AD-A093 887 AIR FORCE WRIGHT AERONAUTICAL LABS WRIGHT-PATTERSON AFB OH F/G 9/2
SOFE: A GENERALIZED DIGITAL SIMULATION FOR OPTIMAL FILTER EVALU--ETC(U)
OCT 80 S H MUSICK
UNCLASSIFIED AFWAL-TR-80-1108 NL

1 OF 3
AD
A093887

AFWAL-TR-80-1108

SOFE: A GENERALIZED DIGITAL SIMULATION FOR OPTIMAL FILTER EVALUATION USER'S MANUAL

STANTON H. MUSICK
REFERENCE SYSTEMS BRANCH
SYSTEM AVIONICS DIVISION

OCTOBER 1980

TECHNICAL REPORT AFWAL-TR-80-1108

Final Report for Period 1 January 1976 to 31 July 1980

Approved for public release; distribution unlimited.

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Stanton H. Musick

STANTON H. MUSICK
Project Engineer

William E Shephard

WILLIAM E. SHEPHARD, Tech Mgr
Reference Systems Software Group
Reference Systems Branch

FOR THE COMMANDER

Ronald L Ringo

RONALD L. RINGO
Chief, Reference Systems Branch
System Avionics Division
Avionics Laboratory

Raymond E Siferd

RAYMOND E. SIFERD, Col, USAF
Chief, System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAN*, W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL-TR-80-1108	2. GOVT ACCESSION NO. AD-A093 281	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFE: A Generalized Digital Simulation for Optimal Filter Evaluation, User's Manual		5. TYPE OF REPORT & PERIOD COVERED Final Report for Period 1 Jan 76 - 31 Jul 80
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Stanton H. Musick		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Avionics Laboratory (AFWAL/AARA) AF Wright Aeronautical Laboratories, AFSC Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1206 01 20
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE October 1980
		13. NUMBER OF PAGES 202
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulation	Suboptimal Estimation	Integrated Systems
Monte Carlo Simulation	Kalman Filter	Truth Model
Digital Simulation	Extended Filter	Filter Model
FORTRAN Program	Square Root Filter	Ensemble Average
Optimal Estimation	Performance Analysis	Numerical Integration

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a general-purpose program that was developed to help design and evaluate Kalman filters for integrated systems. The program, which is named SOFE, is a Monte Carlo simulation that can be used for system performance analysis once the Kalman filter is designed and verified. SOFE is written in FORTRAN and is intended for batch use on a digital computer. This report is a complete user's manual for SOFE. It documents the equations used, the program structure, the input requirements and output options, and

next page.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

'92162

concludes with examples of SOFE use in both linear and nonlinear (extended) Kalman filter design studies.

SOFE is divided into two modules named basic SOFE and user-written SOFE. Basic SOFE accomplishes all of the usual bookkeeping and integration functions of a Monte Carlo simulation (see below) and, in addition, provides the algorithm for measurement update of the user's filter. As its name implies, user-written SOFE is a module of FORTRAN-coded subroutines supplied by the user. These routines specify the system under study, including both truth and filter components. The rationale for this division, which places the constant routines in basic SOFE and the variable routines in user-written SOFE, is to free the designer to concentrate on more central issues including problem definition, model specification, model validation, and insight acceleration through experimentation.

The 31 routines of basic SOFE perform I/O (including printer plots), problem setup, run setup, numerical integration of the ordinary differential equations that specify the system dynamics, measurement update, run termination and problem termination. Basic SOFE also provides for consecutive runs that, taken together, constitute a study. User-written SOFE must have nine FORTRAN routines that supply derivatives, measurements, truth model fluctuations, trajectory data, etc.

SOFE is designed to be efficient in its use of core and time. Savings were obtained in these areas by sparse matrix methods, by storing only the nonredundant portions of symmetric matrices, by packing all vectors and matrices in juxtaposition in a single array, by avoiding zero multiplies, and by elimination of double subscripts.

SOFE was developed on a CDC CYBER-74 computer where it compiles in eleven seconds and uses 74000 octal words of memory to solve a small problem. Some effort was made to adhere to ANSI constructs but high portability is not claimed. A companion post-processor program (SOFEPL) is available for doing ensemble averaging across runs and then making various pen plots.

FOREWORD

This report was written by Stanton H. Musick of the Reference Systems Branch, System Avionics Division, Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Wright Patterson AFB, Ohio.

The work documented herein was carried out under Project Work Unit 1206 0120. Most of this work occurred in the period January to May 1978 and was documented in draft form in June 1978 in AFAL-TM-78-19.

Since June 1978, work has continued on 1206 0120 to refine both the SOFE program and its documentation, contained herein. This report constitutes a portion of the final documentation for this work unit. A companion report on SOFEPL, Reference 5, documents a postprocessor for SOFE capable of making line plots.

The author would like to acknowledge three people for their assistance in developing, testing and documenting both SOFE and SOFEPL. Nelson Estes, who was involved early in the day-to-day work on the SOFE code, wrote several I/O routines and helped implement the external trajectory capability. Richard Feldmann, who replaced Nelson during SOFE development, modified the sparse matrix processing routines

to simplify structure and increase efficiency, helped maintain the program through numerous revisions, designed and built the plot postprocessor SOFEPL, and helped author the SOFEPL report, Reference 5. Elizabeth Ditmer put the manuscript version of this report in final printed form using a Digital Equipment Corporation (DEC) program named RUNOFF. Most text preparation and all line justification, paragraph spacing, table setup, etc., were accomplished under RUNOFF while most figures were made using a TEKTRONIX 4014 graphics terminal interacting with an in-house program called FLOWCHART, which ran on a DEC PDP 11/40 computer. My special thanks to all of you, Nelson, Dick and Libby.

CONTENTS

CONTENTS(con't)

FIGURES

TABLES

SYMBOLS

$\triangleq$	Equals by definition
$\cong$	Equals approximately
$\underline{(\ )}$	Column vector
$\dot{(\ )}$	Time derivative
$\hat{(\ )}$	Estimated or computed value (not the true value)
$\tilde{(\ )}$	Measured value
$(\ )^{-1}$	Matrix inverse
$(\ )^{T}$	Matrix or vector transpose
*	Multiplication (not convolution)
**	Exponentiation
I	Identity matrix
E{ }	Expected value of { }
ti^{-}	Argument at time ti, before measurement incorporation
ti^{+}	Argument at time ti, after measurement incorporation
ti^{+c}	Argument at time ti, after measurement incorporation and feedback correction
$\underline{(\ )}^{-}$	Vector at time ti, before measurement incorporation
$\underline{(\ )}^{+}$	Vector at time ti, after measurement incorporation
$\underline{(\ )}^{+c}$	Vector at time ti, after measurement incorporation and feedback correction

ABBREVIATIONS

ANSI	American National Standards Institute
CDC	Control Data Corporation
DEC	Digital Equipment Corporation
D.E.	differential equation
EOF	end-of-file
INS	inertial navigation system
I/O	input/output
SOFE	Simulation for Optimal Filter Evaluation (pronounced 'so-fee')
SOFEPL	SOFE PLot program

1.0 INTRODUCTION

SOFE is a Monte Carlo simulation program that was developed to help analyze integrated systems that employ Kalman filter estimation techniques. SOFE should be useful in all phases of most filter analysis projects, beginning with the initial filter design and ending with a full system performance analysis.

This is the SOFE User's Manual. It is written primarily for people who already understand Kalman estimation. It should help them shorten the cycle from filter design through filter verification to performance analysis by removing much of the mundane programming load and providing a broad range of options for viewing performance results. One such option is provided by an ensemble averaging and plotting program (SOFEPL) which is documented in a companion volume to this one, Reference 5.

This section begins with a background discussion that shows the need for a program such as SOFE and compares the Monte Carlo to the covariance analysis approach. A SOFE overview then follows to relate the purpose, evolution and nature of the program. A brief section-by-section review of this report concludes this section.

1.1 Background

A large class of modern problems requires the estimation (and control) of the state of a dynamic system based on noise-corrupted observations. These problems arise in a variety of fields including physics, economics, medicine and engineering. When the dynamic system is continuous and the observations discrete, the solution to the estimation part of such problems is given by an algorithm called the continuous-discrete Kalman filter.

A Kalman filter is an optimal, recursive, computational algorithm that is used to combine functionally-related measurements in order to estimate desired variables. The filter design procedure begins with the development of mathematical and statistical models to describe the truth system including system and measurement dynamics, system disturbances and measurement errors, and initial condition information. These truth models are often both complex and large and must usually be simplified and reduced in size for implementation in an operational Kalman filter. As in any formulation of mathematical models for physical processes, the challenge is to develop models for the filter that are sufficiently complete to represent the physical phenomenon of interest but not so complex as to be computationally intractable. Thus the designer must build a 'reduced-order' filter model that is simpler than his truth model. This reduced-order filter

will execute faster and use less core than the truth model would have, advantages which are crucial in many applications. If properly designed, the filter will compensate for the mismodeling errors and track the physical phenomenon well enough to satisfy the performance criterion. If improperly designed, the filter state will diverge from the truth state.

In order to develop the required compensation, to select filter states, to test modeling alternatives, to study the effects of uncertainties in the truth model, to test violations of assumptions, to do performance tradeoffs, to provide problem insight, and so forth, computer programs are required. Two types of programs are commonly used, the simulation and the covariance analysis. This document is a user's manual for a simulation named 'SOFE' (1).

Simulation is a much used procedure whereby one constructs an experiment on the computer that emulates the truth system (the environment) and the filter system operating together. The mathematical models for describing the dynamics of both the filter and truth systems are (possibly nonlinear) stochastic differential equations. In addition, (possibly nonlinear) stochastic algebraic equations model the measurements. The random driving functions for the

(1)Simulation for Optimal Filter Evaluation, pronounced so-fee.

truth system as well as the noisy measurements are simulated with the aid of random number generation. These measurements are processed by the Kalman algorithm to produce updated estimates of the filter state and its covariance. Between measurements, the filter state and covariance are propagated by numerical integration. By making repeated runs using different random number sequences, one can form the appropriate ensemble statistics to determine whether the candidate filter design diverges or tracks the truth system within acceptable bounds.

A covariance analysis generates the second-order statistics for both the truth model and the filter design directly so that one covariance analysis run is equivalent to an ensemble of Monte Carlo simulation runs. The potential for computer savings is obvious but the technique only works under several strict assumptions, principal among which are linearity in all models and Gaussian random processes. Often covariance analysis is most appropriate in the early phase of a project when these assumptions are tolerated for the purpose of making a preliminary design or performance prediction. A simulation can be used to study many aspects of a filter design problem that covariance analysis cannot handle and is therefore a natural next step in filter development and performance analysis.

1.2 SOFE Overview

SOFE is an efficient, general-purpose, Monte Carlo simulation program for analyzing integrated systems that employ Kalman estimation techniques. SOFE is a simulation in the sense described above, i.e., it provides a means for constructing a truth system and for testing a filter to track that system in a series of computer experiments. The truth system, represented by a model whose state vector is denoted $\underline{X}s$, is described by a set of stochastic differential equations, supplied by the user, that emulate the real world with its attendant random qualities. The filter system is also described by differential equations, again user-supplied. The filter state vector and its error covariance are denoted $\underline{X}f$ and Pf respectively.

SOFE is general-purpose in the sense that all types of filter design problems can be studied. This occurs because the user describes his particular problem through a set of nine subroutines that he writes and appends to the basic SOFE program. No assumptions are made in the basic SOFE program that particularize it to a single problem.

SOFE is efficient in its use of core and computing time. Since Pf is symmetric and S ($Pf=S*S^T$) is upper triangular, it is sufficient to work with only those elements of Pf and S on and above the diagonals. Thus both Pf and S,

which are square matrices of order NF, are stored and addressed as linear arrays of length NF(NF+1)/2. This costs something in programming complexity but yields storage savings for these two arrays approaching 50% in high-order designs. Further core savings are obtained by dense packing of all vectors and matrices in a single array that may be conveniently contracted or expanded to better fit the user's problem. Computing time savings accrue from increased use of singly subscripted arrays and elimination of all zero multiplies in forming the derivative of Pf.

SOFE is an outgrowth of two other programs, GCAP and MCAP, References 1 and 2. GCAP is a covariance analysis program that uses sparse matrix storage and efficient matrix manipulation techniques to save computer space and execution time. SOFE has borrowed and improved on both of these features to achieve similar advantages. MCAP on the other hand is a true Monte Carlo simulation (like SOFE) that was constructed by melding many of the routines from GCAP with a revised executive and some new routines for handling state vector propagation/update. Both GCAP and MCAP have several prominent deficiencies: both use a fixed step integrator that lacks error control; MCAP propagates $\underline{X}s$, $\underline{X}f$ and Pf as if they were independent when in general they are not; both use the standard Kalman filter update equations which can lead to negative covariances; both restrict the size of the

problem that can be worked. SOFE corrects these deficiencies and adds some new capabilities including printer plotting, line (pen) plotting as is done using a line plotter or a graphics terminal, the ability to acquire and interpolate data from an external trajectory tape, and validation of the user's input data. In addition, SOFE formats and limits the amount of output on each printed page, centralizes the control of output in a single routine, provides a standard check number at run conclusion, uses a very compact structure for packing the vectors and matrices of the problem in blank COMMON, and adheres to ANSI constructs insofar as possible.

SOFE is written in FORTRAN using only single precision quantities. It was developed on a Control Data Corporation CYBER 74 computer where it executes in batch mode. It consists of a main program, 29 subprograms and a block data routine which together are called 'basic SOFE'. A complete load module consists of basic SOFE plus nine user-written routines. Basic SOFE loads in 70000 octal words of memory on the CDC system. A sample problem consisting of 9 truth states and 5 filter states (Section 5.1) boosts the total memory requirement to 74000 octal words.

A reasonable effort has been made to use conventions, ANSI standard constructs and modular concepts that will ex-

pedite SOFE's transfer to other machines. SOFE uses several routines from the standard FORTRAN math library (SQRT, ABS, AMIN1, etc.) plus five special CDC library routines: DATE, TIME, EOF, RANSET and RANF. SOFE requires these peripherals: card reader or data input terminal; line printer; eight (local) files for data I/O. In addition, if line plots are desired, the user will need the DISSPLA post-processor software and a computer graphics device.

1.3 Scope

This report is a user's manual for SOFE. We will try to acquaint the user with all its attributes and limitations and, by example, show him how it operates. However, this will not be a training manual for Kalman filter design. Such an effort is considerably beyond the scope of this report.

Section 2 presents the propagation and square root update equations that form the mathematical basis for SOFE's design. Section 3 describes the program, giving information about its structure and coding conventions. Section 4 specifies the format and content of program inputs, outputs and user-written routines. Section 5 presents two example Kalman filter design problems, one linear and one nonlinear. Appendices A, B and C give samples of output from these designs. Appendix D covers job control for attaching, compil-

ing, loading and executing SOFE on the CDC computer under the NOS/BE operating system.

2.0 SIMULATION MATHEMATICS

The mathematics underlying the SOFE simulation consists of general models for the truth and filter systems together with algorithms for the extended Kalman filter. All of these models and algorithms are presented in this section, along with some discussion of numerical techniques that are used in the basic SOFE code. The section emphasizes what 'data' the user must supply, via his own code, to effect a solution of the Kalman algorithms.

We assume physical processes that are inherently continuous and measurements that are discrete, a combination that leads to the so-called 'continuous discrete Kalman filter'. Purely continuous systems or other discrete-continuous combinations are not addressed. Since the literature abounds with good material on the Kalman filter equations, e.g. Reference 6, we present them here with some explanation but without derivation.

A DEC program named RUNOFF was used to produce this document on a computer printer. Since RUNOFF cannot generate subscripts or superscripts, some compromises must be used, especially when writing equations. In particular, we chose to use trailing letters instead of subscripts (e.g. ti instead of t_i) and to insert superscripts manually

(e.g. in $\hat{\underline{X}}$f the ^ is so entered). Also, only one Greek letter is used (a concocted θ) and special symbols such as overbars are employed sparingly.

Note that a trailing 's' on a symbol (e.g. $\underline{X}$s, $\underline{w}$s, $\underline{v}$s) marks that symbol as a truth model quantity. Similarly, a trailing 'f' on a symbol (e.g. $\underline{X}$f, Qf, $\underline{h}$f) marks it as a filter model quantity. 's' was used instead of 't' for truth model quantities because 't' is reserved for time. The s-quantities may be thought of as <u>s</u>tandards against which f-quantities will be compared.

To distinguish between matrices, vectors and scalars, matrices are denoted by a leading upper-case letter that is not underlined, e.g. F, H, Qs. Vectors are denoted by upper- or lower-case letters and are always underlined, e.g. $\underline{X}$s, $\underline{w}$s, $\underline{f}$(.). Scalars are either upper- or lower-case and are not underlined, e.g. i, j, t, Ai. This scheme allows one to recognize a vector immediately by its underline, but forces one to distinguish a matrix from an uppercase scalar by the context of its use.

2.1 <u>Truth Model</u>

A quite general representation of a dynamic, continuous-time, physical system is given by the following vector, stochastic, ordinary differential equation:

$$\dot{\underline{X}}s(t) = \underline{g}(\underline{X}s,t) + \underline{w}s(t) \tag{2-1}$$

where

t is time

$\underline{X}s(t)$ is the truth system state vector (NSx1)

$\underline{g}(.)$ is the truth system dynamics vector (NSx1)

$\underline{w}s(t)$ is a zero-mean white Gaussian random process of dimension NSx1 with $E\{\underline{w}s(t)\underline{w}s^T(t+T)\} = Qs(t)*\delta(T)$

Qs(t) is the truth system noise strength (NSxNS)

and where, for an initial time to, $\underline{X}s(to)$ is a random vector, independent of $\underline{w}s(t)$, distributed as a zero-mean Gaussian variable. This initial value is denoted $\underline{X}so$. In the above, $\delta(T)$ is the Dirac delta function and E is the expected value operator.

The discrete-time vector output $\underline{Z}s(ti)$ for the system represented by (2-1) is modeled as

$$\underline{Z}s(ti) = \underline{h}s(\underline{X}s,ti) + \underline{v}s(ti) \tag{2-2}$$

where

ti is a discrete measurement time, i=1,2,3,...

$\underline{Z}s(ti)$ is the measurement vector (Mx1)

$\underline{h}s(.)$ is the measurement model function (Mx1)

$\underline{v}s(ti)$ is an Mx1 zero-mean white Gaussian random sequence independent of both $\underline{w}s(t)$ and $\underline{X}so$ with

$$E\{\underline{v}s(ti)\underline{v}s^T(tj)\} = Rs(ti)*\delta ij$$

where Rs is the MxM measurement noise matrix and δij is the Kronecker delta function

Together equations (2-1) and (2-2) form the most detailed model of the truth (meaning actual) system. Since most physical systems of any complexity are nonlinear in nature, both functions $\underline{g}(\underline{X}s,t)$ and $\underline{h}s(\underline{X}s,ti)$ are potentially nonlinear in $\underline{X}s$, a fact connoted here by the inclusion of $\underline{X}s$ in the argument lists of $\underline{g}(.)$ and $\underline{h}s(.)$. $\underline{X}s$ is the physical process that the filter will attempt to track. SOFE solves the differential equation in (2-1) in two stages.

In the first stage the homogeneous part of (2-1), namely $d\underline{X}s/dt = \underline{g}(\underline{X}s,t)$, is propagated over a time interval (DTNOYS) specified by the user. This propagation occurs by means of a fifth-order numerical integrator provided in SOFE. The user supplies the function $\underline{g}(.)$ in subroutine XSDOT, and the numerical integration occurs automatically.

In the second stage, propagation has just concluded and the accumulated effect of $\underline{w}s(t)$ must be accounted for. A typical method for doing this begins by computing the following 'delta-covariance' matrix:

$$Qd(tj) = Qs(tj)*DTNOYS \qquad (2-3)$$

If DTNOYS is small compared to the Shannon sampling period in $\underline{g}(.)$, Qd(tj) approximates the growth in covariance of the $\underline{X}$s process caused by the random system disturbance $\underline{w}$s(t) on the interval DTNOYS from tj to tj+1. Random noise is injected in $\underline{X}$s by generating a multivariate Gaussian sample $\underline{w}$d(tj) having covariance Qd(tj), and then adding this sample directly to $\underline{X}$s. These actions occur in SNOYS, a user-written subroutine that SOFE calls at DTNOYS intervals. Function subroutine GAUSS is provided in basic SOFE for generating uncorrelated random Gaussian samples of specified mean and variance. Correlated random Gaussian samples can also be generated (for Qd(tj) nondiagonal) by using basic SOFE subroutines PSQRT and GAUSS in conjunction with one another (e.g. see [6], p. 408, problem 7.14).

In some situations the approximation in (2-3) is inefficient and/or inaccurate. Inefficiencies occur when the Shannon sampling period of $\underline{g}(.)$ varies significantly during a run. In this situation DTNOYS, which is fixed, must be set small to accommodate the shortest sampling period in $\underline{g}(.)$. But a small DTNOYS forces a large number of interruptions in the integration process and raises the computation time. Attempting to correct the problem by simply enlarging DTNOYS without changing (2-3) will eventually lead to inaccurate realizations of the $\underline{X}$s process. In such situations other options should be considered for computing Qd(tj).

If $\underline{g}(.)$ is linear in $\underline{X}s$, several options surface immediately. In this case (2-1) is written

$$\dot{\underline{X}}s(t) = G(t)\underline{X}s(t) + \underline{w}s(t)$$

With linear dynamics, methods for improving the numerical approximation in (2-3) are available (e.g. [6], Subsection 6.11) and can be used effectively in a practical filter implementation. For a simulation truth model, however, the exact representation for Qd(tj) is of greater interest. For the linear case, Qd(tj) may be computed exactly by solving the following ordinary differential equation for $\bar{Q}(t,tj)$.

$$\dot{\bar{Q}}(t,tj) \triangleq G(t)\bar{Q}(t,tj) + \bar{Q}(t,tj)G^T(t) + Qs(t)$$
$$\bar{Q}(tj,tj) \triangleq 0$$
$$Qd(tj) = \bar{Q}(tj+1,tj)$$

To implement this solution in SOFE, one would include $\bar{Q}(t,tj)$ in the state vector $\underline{X}s$ and use SNOYS to formulate $\underline{w}d(tj)$ from Qd(tj) as described above. Before exiting from SNOYS, $\bar{Q}(tj+1,tj)$ would be reset to zero to initialize the next integration from tj+1 to tj+2. Operating in this manner will allow DTNOYS to be set larger than if (2-3) is used with assurances that dynamics G(t) and plant noise $\underline{w}s(t)$ are coupled accurately. As a practical matter, however, this approach is significantly more complex than

(2-3) so it may not often be worth the effort.

If $\underline{g}(.)$ is truly nonlinear in $\underline{X}s$, the problem of a more efficient computation for Qd(tj) than (2-3) is even more complex, involving stochastic integrals of products of $\underline{g}(.)$ and $\underline{w}s(t)$, and is well beyond the scope of what can be discussed here.

The knowledgeable reader will note that $\underline{u}s(t)$, the additive deterministic forcing function, is omitted from (2-1). This omission was made in order to simplify the presentation. If $\underline{u}s(t)$ were present in the user's problem, it would be accounted for like $\underline{g}(.)$ is, namely by including it as an additional term in the derivatives specified in user-subroutine XSDOT. Basic SOFE would not need alteration. Similar statements apply for the omission of $\underline{u}f(t)$ from the filter state differential equation, (2-4), to be discussed next.

2.2 Filter Model

The continuous-time physical system and its discrete-time measurement output are modeled for the filter by these two (possibly nonlinear) equations:

$$\dot{\underline{X}}f(t) = \underline{f}(\underline{X}f,t) + \underline{w}f(t) \quad (2\text{-}4)$$

$$\underline{Z}f(ti) = \underline{h}f(\underline{X}f,ti) + \underline{v}f(ti) \quad (2\text{-}5)$$

where

t and ti are defined in (2-1) and (2-2)

$\underline{X}f(t)$ is the filter state vector (NFx1)

$\underline{f}(.)$ is the filter dynamics vector (NFx1)

$\underline{w}f(t)$ is a zero-mean white Gaussian random process of dimension NFx1 with

$E\{\underline{w}f(t)\underline{w}f^{T}(t+T)\} = Qf(t)*\delta(T)$

Qf(t) is the filter noise strength (NFxNF)

$\underline{Z}f(ti)$ is the measurement vector (Mx1)

$\underline{h}f(.)$ is the measurement model function (Mx1)

$\underline{v}f(ti)$ is an Mx1 zero-mean white Gaussian random sequence independent of $\underline{w}f(t)$ with

$E\{\underline{v}f(ti)\underline{v}f^{T}(tj)\} = Rf(ti)*\delta ij$

where Rf is the MxM measurement noise matrix

and where $\underline{X}f(to)$ is a zero-mean Gaussian random vector, independent of both $\underline{w}f(t)$ and $\underline{v}f(ti)$, and denoted $\underline{X}fo$.

In general, (2-4) and (2-5) for the filter are not identical to (2-1) and (2-2) for the truth because of the necessity to construct the filter as a reduced-order system suitable to real-time solution. Thus NF is usually less than NS, $\underline{f}(.) \neq \underline{g}(.)$ and $\underline{h}f(.) \neq \underline{h}s(.)$. Obviously the white Gaussian noise terms are not equal, filter to truth, and their strengths may not be matched either, i.e. Rf may not match Rs nor Qf match Qs.

Note that in (2-4) and (2-5) we are not yet dealing with filter estimates or actual measurements but with underlying models that will eventually lead us to estimates based on measurements.

2.3 Conventional Kalman Filter Equation Summary

The discrete Kalman filter is a recursive data processing algorithm usually implemented in software on a digital computer. At update time, it combines available measurements plus prior knowledge about the system and the measuring devices to produce an estimate of the state $\underline{X}f$ in such a manner that the mean square error is minimized statistically. During propagation, it advances the estimate in such a way as to again maintain optimality.

The coventional Kalman filter performs the above tasks for <u>linear systems</u> and <u>linear measurements</u> in which the driving and measurement noises are assumed to be mutually uncorrelated, white, zero-mean and Gaussian, and the initial conditions are independent, zero-mean and Gaussian. These are precisely the assumptions made for the filter model given by equations (2-4) and (2-5), except that $\underline{f}(.)$ and $\underline{h}f(.)$ may not be linear in $\underline{X}f$. When the system dynamics and measurement relationships are linear in $\underline{X}f$, (2-4) and (2-5) can be rewritten as

$$\dot{\underline{X}}f(t) = F(t)\underline{X}f(t) + \underline{w}f(t) \tag{2-6}$$

$$\underline{Z}f(ti) = H(ti)\underline{X}f(ti) + \underline{v}f(ti) \tag{2-7}$$

where the assumptions regarding noises and initial conditions remain those listed with (2-4) and (2-5).

Now define $\hat{\underline{X}}f$ as the estimate of $\underline{X}f$, specifically the conditional mean, conditioned on the history of measurements taken up to the present time. The error covariance of $\underline{X}f$, termed Pf, is the expected value of the error in this estimate.

$$Pf = E\{(\underline{X}f-\hat{\underline{X}}f)(\underline{X}f-\hat{\underline{X}}f)^T\} \tag{2-8}$$

The Kalman estimation equations appropriate for the system in (2-6) and (2-7) are summarized in Figure 2-1. Note that the superscripts - and + on $\hat{\underline{X}}f$ and Pf refer respectively to before and after measurement incorporation at ti. Also note that the tilde ˜ over $\underline{Z}s$ in (2-12) denotes a realized value from the measurement truth model.

TIME PROPAGATION BETWEEN UPDATES

Some form of integration, usually a numerical approximation, is used to solve these ordinary D.E.s on the interval (t_{i-1}, t_i) between updates.

$$\dot{\hat{X}}_f(t) = F(t)\hat{X}_f(t) \tag{2-9}$$

$$\dot{P}_f(t) = F(t)P_f(t) + P_f(t)F^T(t) + Q_f(t) \tag{2-10}$$

MEASUREMENT UPDATE AT t_i

At measurement time t_i, the estimate is updated using these algebraic relationships.

$$K = P_f^- H^T (H P_f^- H^T + R_f)^{-1} \tag{2-11}$$

$$\hat{X}_f^+ = \hat{X}_f^- + K(\tilde{Z}_s - H\hat{X}_f^-) \tag{2-12}$$

$$P_f^+ = P_f^- - KHP_f^- \tag{2-13}$$

INITIAL CONDITIONS AT t_o

$$\hat{X}_{fo} \triangleq E(X_f(t_o)) \tag{2-14}$$

$$P_{fo} \triangleq E((X_f(t_o)-\hat{X}_{fo})(X_f(t_o)-\hat{X}_{fo})^T) \tag{2-15}$$

Figure 2-1. Conventional Kalman Filter Equations

2.4 Extended Kalman Filter Formulation

The extended Kalman filter is a variation of the conventional filter which relaxes the requirement that the system and measurements be linear. It is the filter generally used in practice for nonlinear applications. This subsection presents the extended filter equations as a logical extension of the conventional equations.

For $\underline{f}(.)$ or $\underline{h}f(.)$ nonlinear in $\underline{X}f$, define these partial derivatives

$$F(t;\underline{X}f) \triangleq \partial\underline{f}(\underline{X}f,t)/\partial\underline{X}f \tag{2-16}$$

$$H(ti;\underline{X}f) \triangleq \partial\underline{h}f(\underline{X}f,ti)/\partial\underline{X}f \tag{2-17}$$

where the differentiation is 'row-type' meaning that the derivative of a scalar with respect to a column vector is a row vector. This produces dimensions for F(.) and H(.) of NFxNF and MxNF respectively. F(.) and H(.) may be viewed as sensitivity matrices that relate small perturbations in $\underline{X}f$ to changes in $\dot{\underline{X}}f$ and $\underline{Z}f$ as in the differential calculus. F(.) is called the 'filter dynamics partial matrix' and H(.) the 'measurement sensitivity matrix'. Define the perturbation $\underline{DX}$ of $\underline{X}f$ from its current estimate $\hat{\underline{X}}f$.

$$\underline{DX} \triangleq \underline{X}f - \hat{\underline{X}}f \tag{2-18}$$

The perturbation $\underline{DX}$ is called the error state while $\underline{X}f$ is the full state.

Expand $\dot{\underline{X}}f$ and $\underline{Z}f$ from (2-4) and (2-5) in Taylor series expansions about $\hat{\underline{X}}f$ in powers of $\underline{DX}$. After truncating $\underline{DX}*\underline{DX}$ and all higher powers of $\underline{DX}$ from the resulting expansions, one arrives at the following linearized perturbation equations in $\underline{DX}$.

$$\dot{\underline{DX}}(t) = F(t;\underline{X}f)\underline{DX}(t) + \underline{w}f(t) \quad (2\text{-}19)$$

$$\underline{DZ}(ti) = H(ti;\underline{X}f)\underline{DX}(ti) + \underline{v}f(ti) \quad (2\text{-}20)$$

In (2-19) and (2-20) we have equations that meet the assumptions of the conventional filter. Thus, a direct estimate $\hat{\underline{DX}}^{+}$ of the error state $\underline{DX}$ can be made from measurements $\tilde{\underline{DZ}}(ti)$ using equations (2-11) through (2-13). The measurement difference $\tilde{\underline{DZ}}(ti)$ is called the residual. It is formed in this case by subtracting the actual ($\tilde{\underline{Z}}s$) and predicted ($\hat{\underline{Z}}f$) measurements.

$$\tilde{\underline{DZ}}(ti) = \tilde{\underline{Z}}s(ti) - \hat{\underline{Z}}f(ti) \quad (2\text{-}21)$$

where

$$\tilde{\underline{Z}}s(ti) = \underline{h}s(\underline{X}s,ti) + \underline{v}s(ti) \quad (2\text{-}22)$$

$$\hat{\underline{Z}}f(ti) = \widehat{\underline{h}f(\underline{X}f,ti)}$$

$$\approx \underline{h}f(\hat{\underline{X}}f,ti) \quad (2\text{-}23)$$

With $\underline{\hat{DX}}^+$ in hand, (2-18) can be turned around to yield an updated full-state vector.

$$\underline{\hat{X}}f^+ = \underline{\hat{X}}f^- + \underline{\hat{DX}}^+ \tag{2-24}$$

Equation (2-24) folds all the available data into a single full-state estimate and thereby allows $\underline{\hat{DX}}(ti^+)$ to be reset to zero. Returning to (2-19) and taking the expected value of both sides, we see that, with a zero initial condition, $\underline{\hat{DX}}(t)$ will be zero over the entire interval between updates.

$$\underline{\hat{DX}}(t) = \underline{0} \qquad \text{for } ti^+ \leq t \leq ti_{+1}^- \tag{2-25}$$

With $\underline{\hat{DX}}(ti^-)$ zero, the error-state update equation (based on (2-12)) simplifies to $\underline{\hat{DX}}^+ = K\underline{\tilde{DZ}}$, which on substitution in the full-state update equation (2-24) produces

$$\underline{\hat{X}}f(ti^+) = \underline{\hat{X}}f(ti^-) + K(ti)\underline{\tilde{DZ}}(ti) \tag{2-26}$$

where $\underline{\tilde{DZ}}(ti)$ is given by (2-21).

To obtain an equation for the propagation of $\underline{\hat{X}}f$ between updates, take the expectation of Taylor's expansion of (2-4) retaining only the first term.

$$\begin{aligned} \dot{\underline{\hat{X}}}f(t) &= \widehat{\underline{f}(\underline{X}f,t)} \\ &\cong \underline{f}(\underline{\hat{X}}f,t) \end{aligned} \tag{2-27}$$

Note that the form of $\underline{f}(.)$ in (2-27) is identical to that in (2-4) so that none of the dynamic nonlinearities are lost during propagation.

In (2-27) we have arrived at the desired equation for propagation of $\hat{\underline{X}}f$. Derivatives $\dot{\hat{\underline{X}}}f$ are supplied to SOFE through user-routine XFDOT, and SOFE's fifth-order numerical integrator solves (2-27) for $\hat{\underline{X}}f$ with initial conditions after each update being those produced by (2-26). Equation (2-26) not only updates the full state vector but, in effect, relinearizes the state around a new nominal that enhances the validity of (2-19) and (2-20) for the next propagation. Although (2-26) could serve as written, it will be replaced by a numerically superior method as discussed in the next subsection.

Consider now the question of the error covariance Pf of the full-state $\underline{X}f$ and its relationship to the error covariance Pfd of $\underline{DX}$.

$$
\begin{aligned}
Pfd &= E\{(\underline{DX}-\hat{\underline{DX}})(\underline{DX}-\hat{\underline{DX}})^T\} & \\
&= E\{\underline{DX}\ \underline{DX}^T\} & \text{by (2-25)} \\
&= E\{(\underline{Xf}-\underline{Xf})(\underline{Xf}-\underline{Xf})^T\} & \text{by (2-18)} \\
&= Pf & \text{by (2-8)}
\end{aligned}
$$

In words, the error covariance of $\underline{X}f$ is identical to that of

$\underline{DX}$. But $\underline{DX}$'s covariance is given by conventional filter equation (2-10) with $F(t;\hat{\underline{X}}f)$ replacing $F(t)$. Thus

$$\dot{P}f(t) = F(t;\hat{\underline{X}}f)Pf(t) + Pf(t)F^T(t;\hat{\underline{X}}f) + Qf(t) \quad (2\text{-}28)$$

Equation (2-28) governs the evolution of Pf between updates. For measurement update of Pf^-, (2-13) could serve, but a numerically superior algorithm exists and is implemented in SOFE. The next subsection presents that algorithm.

2.5 Square Root Update Algorithm

The measurement update formulation used in SOFE is the sequential square root form developed by Carlson and documented in [3]. Carlson's approach is algebraically equivalent to the standard approach of (2-11) through (2-13) if these standard equations are used in a 'recursive scalar update mode'. In this mode Hj+1 and $\hat{Z}fj+1$, the measurement sensitivity (row) vector and the predicted value for the j+1th of M simultaneous measurements, are computed based on $\hat{\underline{X}}f^{+j}$, the state estimate available after j scalar measurements have been incorporated iteratively. In nonlinear problems this recursive relinearization between measurements yields improved estimates of $\hat{\underline{X}}f^+$ and Pf^+ whether Carlson square root or standard equations are used. However, the Carlson form offers several additional numerical advantages

on finite wordlength computers [6, p 399]:

- o It is numerically stable whereas the standard form is unstable.
- o It is approximately twice as precise as the standard form.
- o It effectively guarantees a nonnegative square root matrix S^+.

The cost of these advantages, a modest increase in computation time and required storage, is judged small compared to their benefit.

This section summarizes the Carlson update equations used in SOFE. Note that the time-propagation equations discussed previously are not those suggested by Carlson in [3]. However, the two time-propagation approaches are alike in the essential fact that both propagate Pf instead of its square root S. For notational simplicity, we suppress time arguments in this subsection.

The error covariance square root matrix S is related to Pf by

$$Pf = S*S^T \tag{2-29}$$

To make S unique, Carlson chooses the upper triangular form

obtained by Cholesky decomposition of Pf. The update process begins by computing S^- from Pf^-, the error covariance available at the end of time propagation. Such an S^- can always be found if Pf^- is positive semidefinite (e.g.[4], p.81). Next, each measurement is processed individually by the update algorithm to produce from the original S^- and $\hat{\underline{X}}f^-$, updated versions denoted $S^+$and $\hat{\underline{X}}f^+$. Finally, S^+ is 'squared' using (2-29) to re-form Pf^+, and the time-propagation procedure is ready to resume. The equations for this sequence are detailed below.

Begin update by computing the Cholesky square root S^-, denoted symbolically as follows:

$$S^- = \sqrt[c]{Pf^-} \tag{2-30}$$

For the details of the computations in (2-30), see Reference 6, page 372. For each realized measurement $\tilde{Z}sj$, $j = 1,2,...,M$, perform the following sequence:

$$\underline{d} = (S^-)^T Hj^T$$

$$Ao = Rfj$$

$$\underline{bo} = \underline{0}$$

$$\text{Repeat for } i = 1 \text{ to } NF \tag{2-31}$$

$$k = i-1$$

$$Ai = Ak + di^2$$

$$\underline{b}i = \underline{b}k + \underline{S}i^{-}di$$

$$\underline{S}i^{+} = (\underline{S}i^{-} - \underline{b}kdi/Ak)(Ak/Ai)^{1/2}$$

$$\hat{\underline{X}}f^{+} = \hat{\underline{X}}f^{-} + (\underline{b}_{NF}/A_{NF})\tilde{D}Zj \tag{2-32}$$

where

Hj = measurement gradient vector, the $j\underline{th}$ row of $H(ti;\hat{\underline{X}}f)$ from (2-17) (2-33)

Rfj = measurement error variance, the $j,j\underline{th}$ element of diagonal Rf (2-34)

di = $i\underline{th}$ element of $\underline{d}$

$\underline{S}i^{-}$ = $i\underline{th}$ column of S^{-}

$\tilde{D}Zj$ = measurement residual, the $j\underline{th}$ element of $\underline{\tilde{DZ}}(ti)$ from (2-21) (2-35)

Where $\hat{\underline{X}}f$ data is required to evaluate Hj or $\tilde{D}Zj$, the estimate through j-1 iterations should be used since it is the best available data.

Note the similarity of the $\hat{\underline{X}}f$ update equation in (2-26) to its replacement (2-32), the gain K for the vector measurement residual $\underline{\tilde{DZ}}(ti)$ corresponding to $\underline{b}_{NF}/A_{NF}$ for the scalar measurement residual $\tilde{D}Zj$. When all M measurements have been processed through (2-31) and (2-32), $\hat{\underline{X}}f$ and S are fully updated and Pf can be re-formed using (2-29).

When the M measurements are uncorrelated, Rf is diagonal and the above procedure goes through directly. When the

M measurements are instead correlated, Rf is not diagonal and the following linear transformation is recommended to provide M uncorrelated measurement combinations $\underline{\tilde{DZ}}'$:

$$V = Rf^{1/2}$$

$$V * \underline{\tilde{DZ}}' = \underline{\tilde{DZ}} \quad \text{--->} \quad \underline{\tilde{DZ}}'$$

$$V * H' = H \quad \text{--->} \quad H'$$

$$R' = I$$

If V is obtained as the Cholesky square root of Rf, then $\underline{\tilde{DZ}}'$ and H' can be obtained by back substitution without inverting V.

2.6 Summary of Extended Filter Equations

Equations (2-27) through (2-32) form the system of equations comprising the extended Kalman filter as implemented in SOFE. These are the equations incorporated in the basic SOFE code. Through his subroutines, the user supplies f(.), F(.), Qf(.), Hj, Rfj, and $\tilde{DZ}j$ to effect the solution of these imbedded equations. A summary of equations (2-27) through (2-32), together with the necessary supporting formulas, is given in Figure 2-2.

Note that SOFE makes no distinction between a linear and a nonlinear problem. During propagation, SOFE only knows it must numerically integrate D.E.s for the truth

model, the filter model and the covariance. It cannot tell linear D.E.s from nonlinear ones. During update, SOFE performs algebraic operations on Pf and $\hat{\underline{X}}f$ using Hj, Rfj and $\widetilde{DZ}j$ data supplied by the user-written subroutine HRZ. It is true that Hj and $\widetilde{DZ}j$ may depend on $\hat{\underline{X}}f$, but this does not alter the form of SOFE's internal algebra. Moreover, since $\widetilde{DZ}j$ is just a scalar residual to (2-32) -- $\widetilde{DZ}j=\tilde{Z}sj-Hj\hat{\underline{X}}f$ for linear problems or $\widetilde{DZ}j=\tilde{Z}sj-\hat{Z}fj$ for nonlinear problems -- the linear/nonlinear problem structures are again masked from view in SOFE. In short, SOFE is equally applicable to linear problems and to nonlinear problems that employ extended filter design principles.

TIME PROPAGATION BETWEEN UPDATES

These D.E.s are propagated using Runge-Kutta type integration.

$$\dot{\hat{X}}f(t) = f(\hat{X}f, t) \quad (2\text{-}27)$$

$$\dot{P}f(t) = F(t;\hat{X}f)Pf(t) + Pf(t)F^{T}(t;\hat{X}f) + Qf(t) \quad (2\text{-}28)$$

where

$$F(t;\hat{X}f) = \partial f(Xf,t)/\partial Xf \,\Big|\, Xf=\hat{X}f \quad (2\text{-}16)$$

MEASUREMENT UPDATE AT t_i

S^- and $\hat{X}f^-$ are updated recursively using these algebraic relationships.

$$S^- = {}^{s}\sqrt{Pf^-} \quad (2\text{-}30)$$

$$\left.\begin{aligned} \hat{X}f^+ &= \hat{X}f^- + (b_{NF}/A_{NF})\tilde{DZ}_J \\ S^+ &= S^- E \end{aligned}\right\} \text{Solved sequentially M times for M measurements } \tilde{DZ}_J \text{ assuming diagonal Rf.} \quad (2\text{-}32)$$

$$Pf^+ = S^+ S^{+T} \quad (2\text{-}29)$$

where

$$d = S_-^{-T} H_J^T \quad (2\text{-}31)$$

$$b_{NF} = S_- d \quad (2\text{-}31)$$

$$A_{NF} = Rf_J + d^T d \quad (2\text{-}31)$$

$$E = {}^{ac}\sqrt{I - dd^T/A_{NF}}$$

ac means 'analytic Cholesky' decomposition. This equation is formal. For the details on updating S, refer to (2-31).

INITIAL CONDITIONS AT t_0 given by (2-14) and (2-15).

Figure 2-2. Extended Kalman Filter as Implemented in SOFE

2.7 Feedback Control

After update, impulsive changes in $\hat{\underline{X}}f^{+}$ and $\underline{X}s$ can be applied as the user desires through subroutine AMEND. These impulsive changes, if used, usually emulate a feedback correction path not directly affected by (2-32). A pure error state formulation of the filter can give rise to the need for such impulsive changes. Another control option is continuous control which was discussed earlier in Subsection 2.1. Control options involving combinations of continuous control and discrete resets may also be implemented. The fact is that the method of control depends on linearity or lack of it, on full state or error state formulation, on accessibility of feedback paths, etc., and is highly problem-dependent. It should be possible to implement most forms of control with the structures already available in SOFE.

2.8 Vector Structure

To avoid the overhead associated with double subscripting, both Pf and S are carried in SOFE as vectors (linear arrays). Since Pf is symmetric and S is upper triangular, only the upper triangular part of each matrix need be saved to preserve its information. The scheme for constructing the appropriate vector from its matrix is to scan the upper triangular part of each matrix columnwise starting with the 1-1 element:

$$\text{Pf-vector} = (p11\ p12\ p22\ p13\ p23\ p33\ p14\ \ldots)^T \tag{2-36}$$

$$\text{S-vector} = (s11\ s12\ s22\ s13\ s23\ s33\ s14\ \ldots)^T \tag{2-37}$$

The size of these vectors is

$$NTR = NF(NF + 1)/2 \tag{2-38}$$

For propagation, the three vectors of interest are $\underline{X}s$, $\hat{\underline{X}}f$ and Pf-vector, which are concatenated into a composite named $\underline{Y}$.

$$\underline{Y} \triangleq \begin{bmatrix} \underline{X}s \\ \hat{\underline{X}}f \\ \text{Pf-vector} \end{bmatrix} \tag{2-39}$$

The derivative $\dot{\underline{Y}}$ is governed by equation (2-1) for $\dot{\underline{X}}s$, (2-27) for $\dot{\hat{\underline{X}}}f$, and (2-28) for $\dot{P}f$. The user supplies $\underline{g}(.)$ for $\dot{\underline{X}}s$ in XSDOT, $\dot{\hat{\underline{X}}}f$ in XFDOT, and F and Qf for $\dot{P}f$ in FQGEN. Of course, $\dot{P}f$ could not be implemented using the straightforward matrix adds and multiplies of (2-28) because of the vector storage mode of Pf. Two special sparse matrix routines were written to form a $\dot{P}$f-vector equivalent to (2-28).

In update, the vectors of interest are $\hat{\underline{X}}f$ and S-vector. In Appendix C of [3], Carlson gives computer algorithms for update of S-vector carried as prescribed in (2-37). His al-

gorithms were used as written.

2.9 Summary

This section has developed equations for propagation of a truth model state $\underline{X}s$, and for propagation and update of a filter model composed of a state $\hat{\underline{X}}f$ and an error covariance Pf. All propagation is accomplished in SOFE via numerical integration using a self-starting, fifth-order, Runge-Kutta type differential equation solver having automatic error control via step-size adjustment. Update is accomplished using the algebraic relationships in (2-29) through (2-32). The operative equations for the truth model are given in Section 2.1 and are summarized for the filter model in Figure 2-2.

3.0 PROGRAM DESCRIPTION

This section presents information about the concept, structure and coding conventions of SOFE. Our goal is to convey the approach and the implementation that were developed for solving the simulation/Kalman estimation problem outlined in Sections 1 and 2.

3.1 Program Concept

SOFE is intended as an efficient general-purpose tool for expediting the construction of a simulation for Kalman filter design and system performance analysis. As such, it carries the flexibility to handle a wide variety of problems without revision in its basic structure. We now discuss some of the strategies that were used to achieve the aforementioned goals.

Examination of filter design simulations shows that tasks performed may be grouped into these eight categories:

- Data I/O
- Problem setup
- Run initialization
- Time propagation
- Measurement update
- Feedback correction
- Run termination
- Problem termination

These eight tasks are organized into the macro-level flow chart in Figure 3-1.

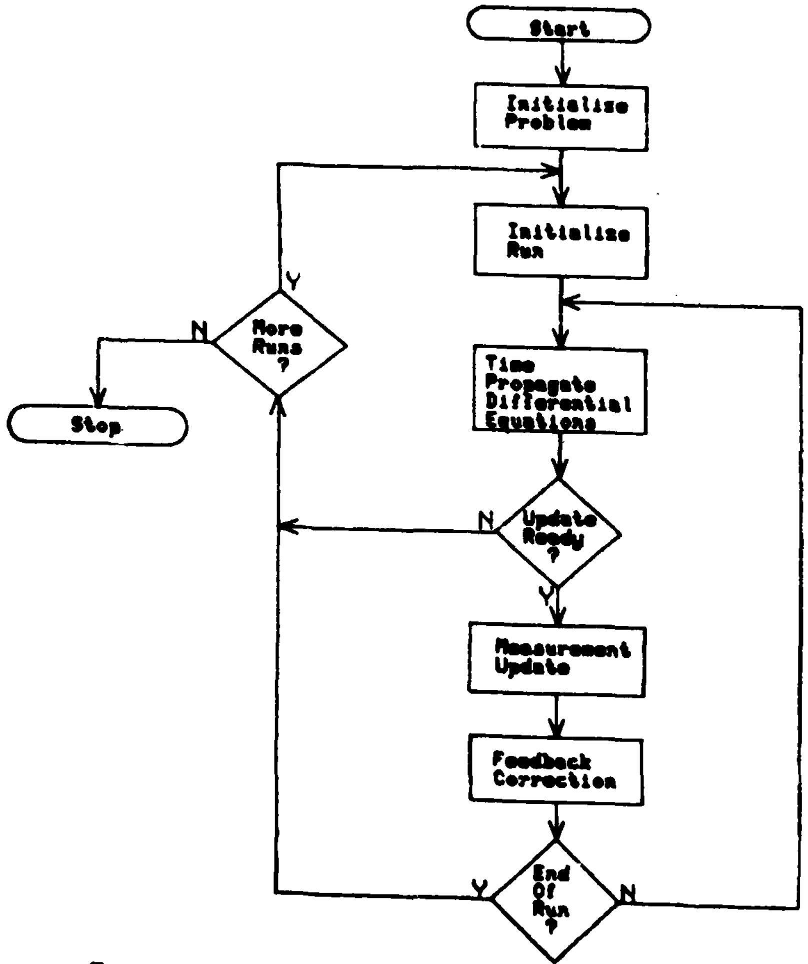


Figure 3-1. Macro Level Flow Chart.

Any filter design simulation must be able to carry out these tasks and SOFE is no different in this regard. SOFE differs in that more of the <u>data</u> for accomplishing these tasks can be user-supplied.

The aforementioned data are of two types, constant and variable. The constant data, which are read into SOFE as card input (Subsection 4.1.1), do not change as the simulation evolves in time. These data contain items such as the dimensions of $\underline{X}s$ and $\hat{\underline{X}}f$, the time intervals between various events, I/O control parameters, etc. These constant data must be supplied to any digital simulation.

The variable data are generated by user-written routines that are called periodically to compute time-varying quantities that bear on the problem solution. Examples of such quantities are derivatives, measurements, truth model fluctuations, trajectory data, etc. These user-written routines give SOFE the flexibility to handle most Kalman filter design studies.

Although user-written routines do provide flexibility, they represent extra time and work to gain access to SOFE. Our constant goal was to keep the user's work to a minimum by doing as much as possible of the repetitious portion of the problem in basic SOFE. This 'principle' produced these interfaces between user code and basic SOFE code.

o Time propagation is accomplished using a (fifth order) numerical integrator in basic SOFE. Derivative values are supplied in user-written subroutines XSDOT, XFDOT and FQGEN.

o Basic SOFE propagates the homogeneous part of $d\underline{X}s/dt$ while the user must inject random noise as an impulse change to $\underline{X}s$ using user-written subroutine SNOYS.

o All update processing of $\hat{\underline{X}}f$ and Pf occurs in basic SOFE but the user must supply H, Rf and Z residual in HRZ.

o The user applies whatever impulsive control he wishes using subroutine AMEND. (Any continuous control would be specified to basic SOFE through the derivatives in XSDOT and XFDOT.)

o Basic SOFE does I/O but USRIN and ESTIX are called from basic SOFE for user-specific input and output respectively.

o Basic SOFE will read and interpolate trajectory data but, if he so desires, the user can construct his own trajectory during execution using user-written TRAJ.

A more detailed picture of the interaction of user-written and basic routines is found in Subsection 3.2

SOFE is designed to be efficient in two areas: use of <u>core</u> and <u>time</u>. Core savings are obtained by the following means:

o Since Pf is symmetric, complete covariance information is retained when only the upper triangular portion (p_{ij}, $i \leq j$) is processed. SOFE propagation, update and I/O algorithms are designed for this upper triangular storage mode. The collected savings in Pf and S total NF(NF-1) words of core.

o Since F and Qf are often sparse matrices, space is provided only for their nonzero values. This requires storing two additional words for the row-column indices of each nonzero element, but usually there is a substantial net saving of core.

o All vectors and matrices needed to solve the user's problem are retained in unlabeled COMMON area A in a dense format. When SOFE needs a particular array, that array is found from its first-word address in A, an address that is assigned at problem setup based on dimensions and sizes specified by the user. Putting all arrays in A allows the user to shrink or enlarge A to fit his problem, a change that can be made by altering just two statements in the main routine (see comments in SOFE).

o The general working space is only NF+2M words, a relatively small number.

Computing time savings are obtained by these means:

o Sparse matrix manipulation methods are used to form the derivative of Pf. Subroutines FPPPFT and ASYSP exploit the sparse nature of F and Qf by eliminating all zero multiplies.

o Elimination of most doubly subscripted arrays in favor of singly subscripted vectors.

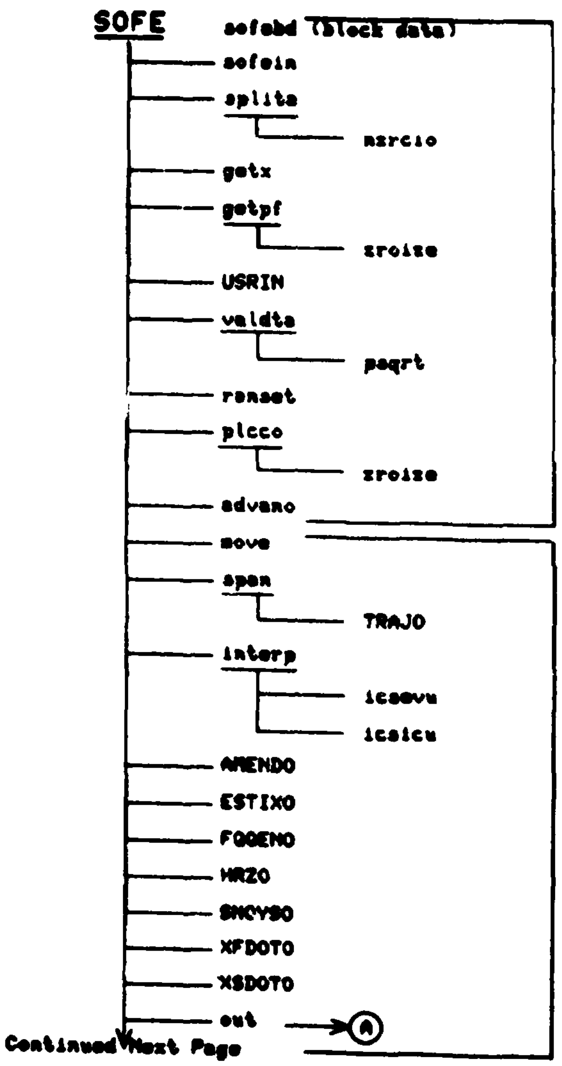
Reference 1 contains further details about most of these efficiency techniques. Note that additional computational savings were obtained in SOFE by switching to an upper triangular storage mode for Pf. The indexing computations are remarkably simpler than those for the lower storage mode used in [1] and [2].

3.2 Program Structure

SOFE is a modular computer program consisting of a main executive, 29 subprograms and a block data routine. SOFE was constructed using a 'top down' approach. Thus it contains a small number of top level, mainly logical routines to provide sequencing and control while the computational algorithms are relegated to lower level routines. This structure is visible in Figure 3-2, a subroutine dependency chart showing, in approximate time order, what calls what. The reader will note the correspondence of the descriptive phrases on the right of Figure 3-2 and the operations in Figure 3-1.

A complete review of program structure would require discussion of individual subroutines. Such a discussion is beyond the scope of this user's manual, but a few notes about the executive structure are needed. Two routines contain almost all the executive functions: SOFE and ADVANS.

SOFE is the main executive. It controls problem and run initialization, measurement update processing, and feedback. ADVANS is in charge of propagation via numerical integration. It schedules all periodic events and forces the integration to pause at each event time so the event action may take place. The six events are:

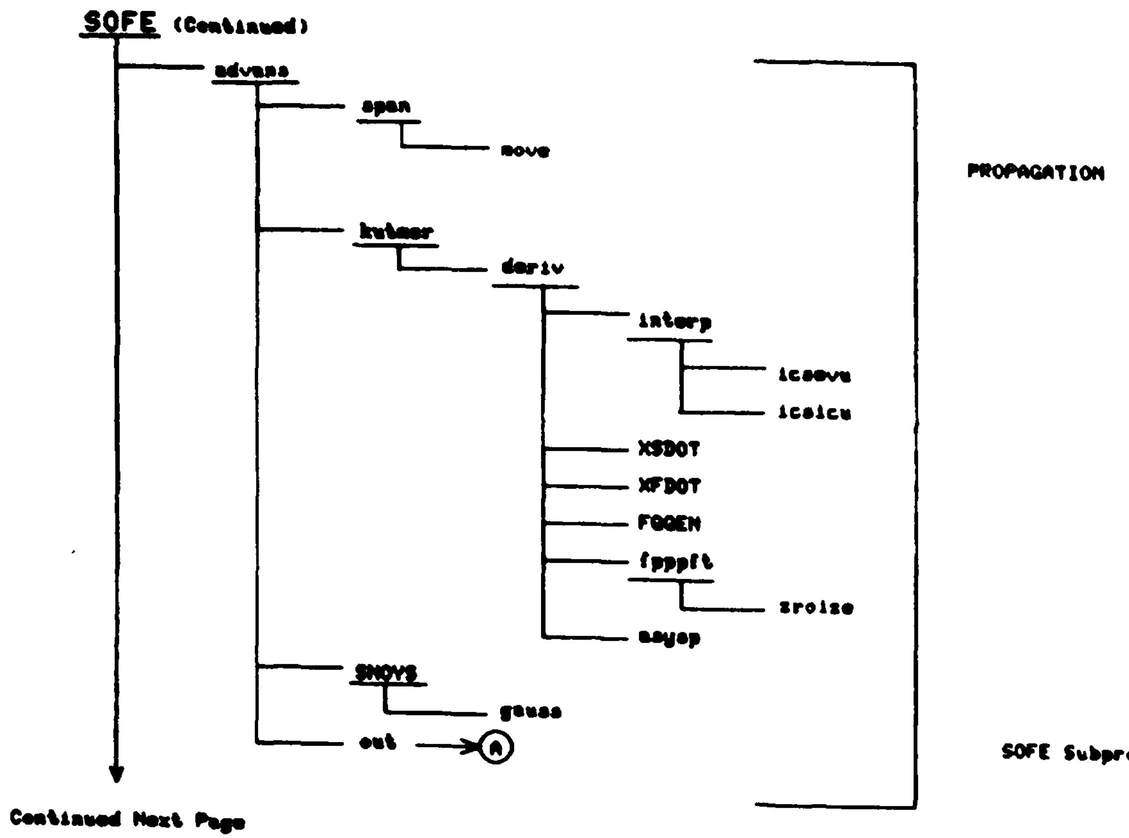


PROBLEM INITIALIZATION

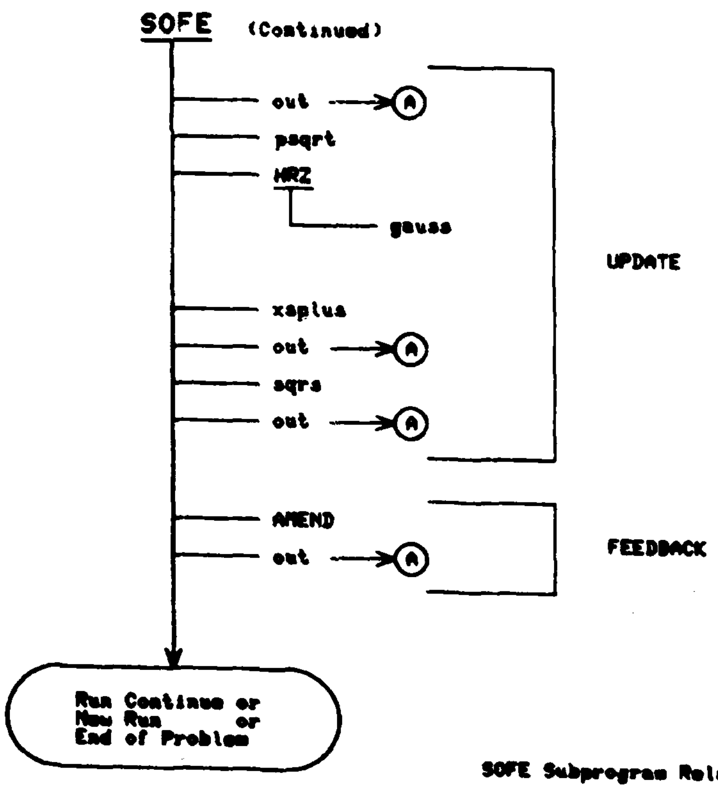
NOTE: Subroutines in lower case letters and the main program 'SOFE' are in basic SOFE. Subroutines in upper case letters are user-written.

RUN INITIALIZATION

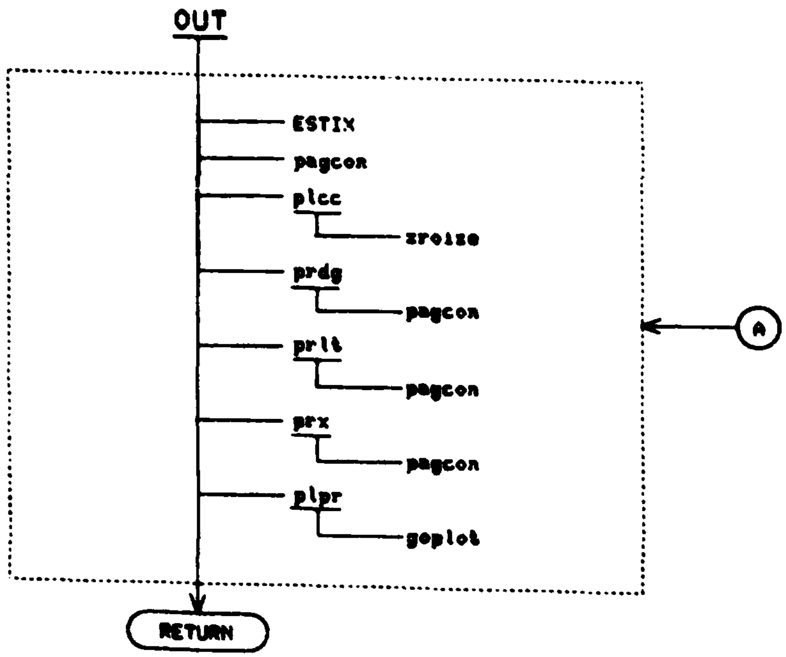
Figure 3-2. SOFE Subprogram Relationships (1 of 4)



SOFE Subprogram Relationships (2 of 4)



SOFE Subprogram Relationships (3 of 4)



SOFE Subprogram Relationships (4 of 4)

- o Update
- o Printed output
- o Calcomp output
- o Printer plot output
- o User output
- o Noise injection in $\underline{X}$s

ADVANS also causes the correct data to be selected from the external trajectory tape (TAPE3) for integration and update purposes.

SOFE uses ten labeled COMMON areas to store data pertinent to its internal workings. Table 3-1 lists these areas and their contents.

Table 3-1

LABELED COMMON DESCRIPTIONS

Area	Contents
DAYTIM	Simulation date and time
DELTAT	Event time intervals
DIFEQ	Integration related quantities
ICOM	A mixture of control and index data beginning with letter I
IPOINT	First word address of all arrays in unlabeled COMMON area A
LCOM	Logical parameters for output control
NCOM	Dimensions and sizes
OTHER	Measurement residual and external trajectory control
TCOM	Times given at input
TITLE	Problem title

The user must not name any COMMON area in his code with one of the above ten names nor should he attempt to use any unlabeled COMMON.

3.3 Coding Conventions

SOFE is written in FORTRAN using mainly 1966 ANSI standard constructs. Exceptions to the ANSI standard rule are:

- o Use of the 'NAMELIST' and 'list directed' conventions for reading input
- o In FORMAT statements, use of special symbols for tabbing and delimiting Hollerith data
- o Use of the ENTRY statement
- o Use of comments following STOPs
- o DATA statements for arrays
- o Use of the ENCODE capability (in GOPLOT only)
- o Use of the octal constant (in GOPLOT only)

Insofar as possible, routines have been kept to a single page for readability purposes. Each routine has a set of comments at its beginning describing its function. Comments are sprinkled throughout the code and considerable effort has been made to make them complete, informative and accurate. The following order was used to list the nonexecutable statements at the beginning of each routine:

- o COMMON
- o DIMENSION
- o EQUIVALENCE
- o EXTERNAL
- o Type
- o DATA
- o NAMELIST

Note that equivalence statements were rarely used. Also

note that only logical variables were 'typed' since the first character default rule for real and integer variables was followed throughout. Required format statements appear following the last return statement.

SOFE uses only single precision variables of the REAL, INTEGER and LOGICAL type. Variables are given meaningful names from the 36 alphanumeric characters (no special symbols).

4.0 SOFE INTERFACES

This section covers input, output and the construction of user-written routines. It begins with an overview that illustrates the flow of input (data and program modules) to the computer and the output of information from the computer.

All input and output in SOFE is accomplished through external data files called tapes. It is usually most convenient for all of these files to reside on disk or magnetic tape, although one file, TAPE5, may be input from cards. SOFE also accepts input from up to two special files, TAPEs 3 and 9. SOFE generates listable output on TAPE6, output for Calcomp plots on TAPE4 and output for problem continuation purposes on TAPE10. Also provided are TAPE8 for user-defined output and TAPE7 for accumulation of printer plot data. These allocations are summarized in Table 4-1.

Table 4-1

SOFE FILE DEFINITION

TAPE3	Input	External trajectory data
TAPE4	Ouput	Data for Calcomp plots
TAPE5	Input	Cards defining user's problem
TAPE6	Output	Listable output device
TAPE7	Temporary	Data for printer plots
TAPE8	Output	User defined
TAPE9	Input	Initial values of $\underline{X}$s,$\underline{X}$f,Pf
TAPE10	Output	Final values of $\underline{X}$s,$\underline{X}$f,Pf

These tapes are sized and ordered on the FORTRAN program card in the main routine of SOFE as follows:

```
PROGRAM SOFE(TAPE5=64/80,TAPE3,TAPE9=512,
            OUTPUT,TAPE4,TAPE6=OUTPUT,TAPE8=512,TAPE10=512,
            TAPE7=512)
```

Note the explicit naming of OUTPUT, the listable output device or printer, the absence of INPUT, and the position of TAPE5. These factors can affect job control which is discussed more fully in Appendix D.

Figure 4-1 shows the flow of SOFE and its data to and from the computer. The solid (dashed) lines that connect to the computer indicate which tapes are mandatory (optional). The flow shown in Figure 4-1 was devised for a CDC computer but would be essentially the same on any computer. Note the special input module called 'user-written routines'. This module contains nine routines that together with the 31 routines of basic SOFE produce a complete program.

4.1 Input

Table 4-1 shows SOFE input on TAPEs 3, 5 and 9. TAPEs 3 and 9 are potentially large files that will usually require magnetic tape or disk storage. TAPE5 is a small file that can be constructed on cards, if desired. TAPEs 3 and 9 are rewound in subroutine SOFEIN during SOFE initialization. TAPE5 is not rewound.

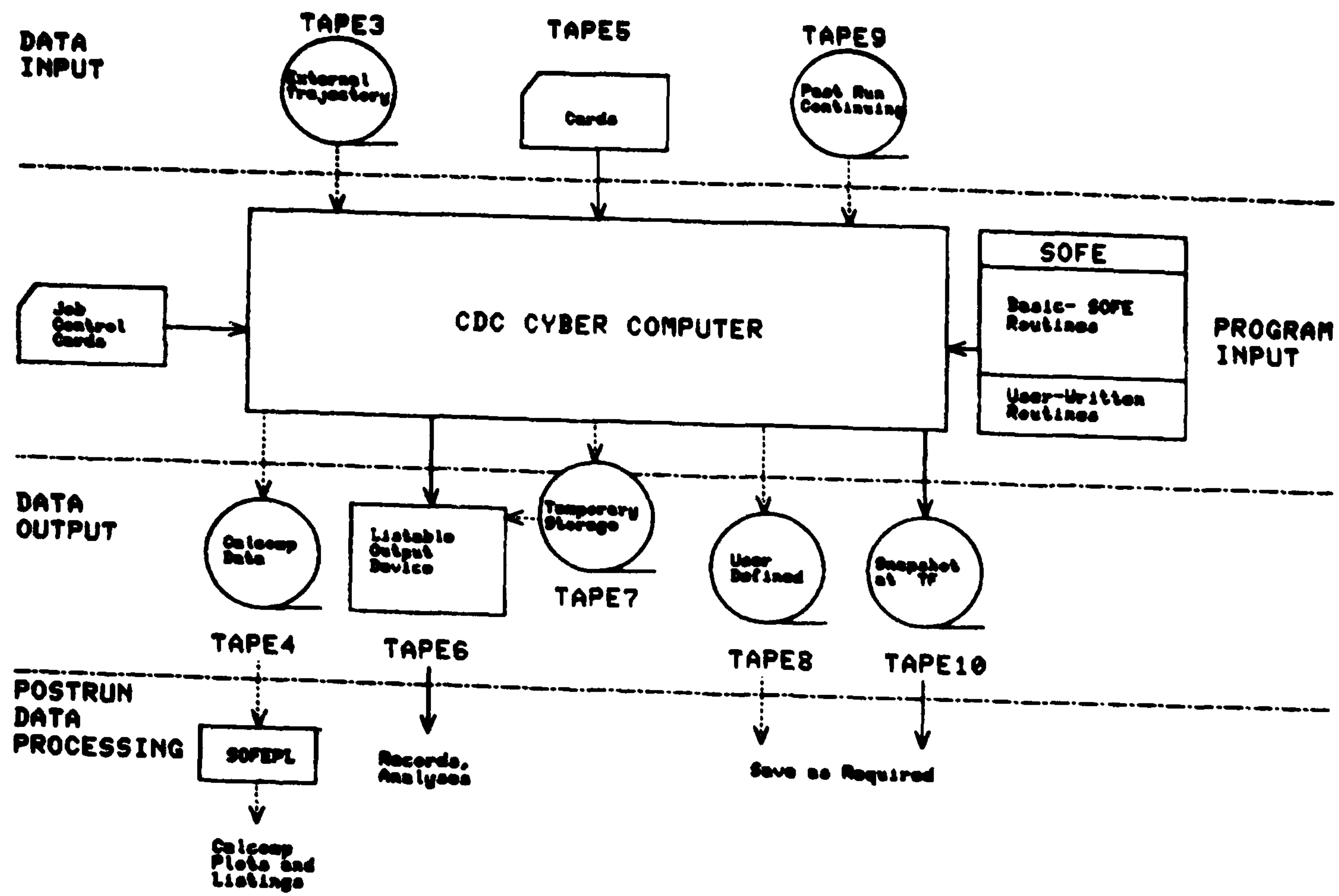


Figure 4-1. Computer Interface Diagram

4.1.1 TAPE5 (Card) Input, Problem Setup and Control

TAPE5 input, which we will often refer to as card input, is read with FORTRAN read statements like the following three:

```
A.  READ (5,100) TITLE
B.  READ (5,*) IROW,ICOL,PDUM
C.  READ (5,PRDATA)
```

Type A is the familiar formatted read. SOFE uses it only for input of 'alpha' data in A format. Type B is a special free-form convention that CDC calls 'list directed'. Data input under this convention must be in order but need not reside in preassigned columns on the card. In addition, multiple quantities may be entered on a single card separated only by commas. Type C is a special convention called NAMELIST which is also of the free-form variety. Taken together, the NAMELIST and list directed conventions form a complete free-form capability. The user should consult the CDC FORTRAN manual for complete details on these conventions. Examples of both are presented herein. Warning to CDC users: avoid the TS and EDITOR compilers because they occasionally choke on list directed input.

Table 4-2 is the ordered list of the quantities to be read from cards, and Figure 4-2 is an example set of card input for the INS problem discussed in Section 5 and Appen-

dix A. Blank lines are inserted between some data sets in Figure 4-2 to enhance readability. They are not required, but they do no harm when used as shown.

The title, in 20A4 format, is the first card. It must be present and may occupy up to one full card (80 columns). The PRDATA namelist, containing problem definition parameters, I/O control flags and integration specifications, follows. If the user desired only defaults, he would enter $PRDATA$. The next section offers a full discussion of each PRDATA parameter. Both the title and the PRDATA list are read from subroutine SOFEIN.

The next card(s) contains the nonzero indices for F. Each index pair is the row-column location of a nonzero element in F. These pairs may be entered in any order so long as the order chosen agrees with that used for nonzero F(i) evaluation in FQGEN (see 4.3.3). The nonzero indices of Qf follow, with the same order convention as F. Since Qf is symmetric, only the indices of nonzero elements on and above the diagonal need be entered. All nonzero indices for both F and Qf are read from NZRCIO.

Table 4-2

SOFE CARD INPUT SEQUENCE

Item	Number	Format	Optional
Problem title	1	20A4	No
PRDATA group	1	Namelist	No (1)
Row-column indices for nonzero elements of F	NZF	List Directed	No (2)
Row-column indices for nonzero elements of Qf	NZQ	List Directed	No (3)
Initial values $\underline{X}so$	NS	List Directed	No (4)
Initial values $\hat{\underline{X}}fo$	NF	List Directed	No (4)
Row index, column index and value of all non-zero elements of Pfo. A 0,0,0. card terminates this input.	$\leq$NF**2	List Directed	No (4)
User defined input as called from USRIN	-	-	Yes
Time-axis scale factor	1	List Directed	Yes (5)
Plot parameter sets A 0,0,0,0. card terminates this input.	$\leq$20	List Directed	Yes (5)
Time-axis title	1	3A10	Yes (5)
Title for ith plot	$\leq$20	8A10	Yes (5)
Y-axis title for ith plot (6)	$\leq$20	3A10	Yes (5)

(1) At least $PRDATA$ must be given, even if only defaults are desired.
(2) These data are omitted if NZF = 0.
(3) These data are omitted if NZQ = 0.
(4) These data are omitted if ICONT = 1.
(5) These data are required only if LPP is TRUE.
(6) Repeat the last two items once for each plot.

```
column 1
GCAP/MCAP SINGLE AXIS INS --- A STANDARD LONG TEST FOR 'SOFE'

 $PRDATA
 NF=5, NS=9, M=2, NZF=7, NZQ=2, NXTJ=1, TF=36000.,
 DTMEAS=30., DTPRNT=3600., DTPRPL=360., DTNOYS=30.,
 LPP=.T.,IPGSIZ=55, $

1,2, 2,3, 2,5, 3,2, 3,4, 4,4, 5,5

4,4, 5,5

9*0.

5*0.

1,1,14400.
2,2,4.
3,3,3.046E-6
4,4,2.350E-15
5,5,4.147E-5
0,0,0.

 $INF TAUF(1)=3600., 300.,
 SDWF(1)=4.848E-8, 6.442E-3,
 RFVCTR(1)=10000.0, 0.25,  $
 $INS TAUS(1)=3600., 300., 3600., 1800.0, 300.0,
 SDWS(1)=4.848E-8, 6.442E-3, 3.22E-3, 3.0E+2, 5.0E-1,
 SDWS0=2.42E-8, $

1.0

1,1,1,1.
2,2,2,1.
3,3,3,3437.75
4,4,4,206265.
5,5,5,1.
0,0,0,0.
  TIME (SECONDS)
      PLOT PARAMETER SET 1,1,1
POSITION *FEET*
      PLOT PARAMETER SET 2,2,2
VELOCITY *FEET PER SECOND*
      PLOT PARAMETER SET 3,3,3
TILT *ARC MINUTES*
      PLOT PARAMETER SET 4,4,4
GYRO DRIFT *DEG PER HR*
      PLOT PARAMETER SET 5,5,5
ACCEL BIAS *FEET PER SECOND2*
```

FIGURE 4-2. SAMPLE OF CARD INPUT

Initial conditions for $\underline{X}s$ and $\hat{\underline{X}}f$, denoted $\underline{X}so$ and $\hat{\underline{X}}fo$, are entered next. There must be NS entries for $\underline{X}so$ and NF for $\hat{\underline{X}}fo$. In the example, repetition factors of 9 (9*0.) and 5 (5*0.) have been used to expedite this input.

The nonzero values of the initial covariance Pfo are entered next. These values are entered in any order by giving their row-column location followed by their numeric value. In the example, Figure 4-2, only one entry per line has been used, but multiple entries are possible if each pair is separated by a slash (see Figure D-2 for an example). Since Pf is symmetric, only nonzero values on and above the diagonal need be specified. Should values below the diagonal be found, their row-column indices are interchanged before normal storage occurs. A card having a nonpositive entry in either the row or column position signals the end of Pfo input.

$\underline{X}so$ and $\hat{\underline{X}}fo$ are read from GETX, while Pfo is read from GETPF. If ICONT is 1, initial state and covariance data are obtained from TAPE9 and should be omitted from cards (see 4.1.1.1).

USRIN input is entered next. It can have any FORTRAN input form, the only requirement being that it reside at this location in the card deck.

The last set of data to be entered from cards is that governing printer plotting. Several printer plot options are available as discussed in 4.1.1.2. The time-axis scale factor is given first, followed by a max of 20 plot parameter sets, one per card. A card whose first three entries are nonpositive signals the end of the plot parameter sets. These plot data are read during SOFE initialization by PLPR. At SOFE conclusion PLPR reads the time-axis title followed by the plot title and Y-axis title for each plot. Note the A formats for these titles in Table 4-2.

4.1.1.1 PRDATA NAMELIST Definitions

Forty parameters are entered through the PRDATA list in CDC NAMELIST format. These parameters, which remain fixed throughout the simulation, specify the user's problem, control I/O, and regulate numerical integration. All parameters have a default value (see Table 4-3) that is invoked in lieu of input data. All numeric parameters are single precision. The following list defines each parameter and gives explanatory data as required. If the parameter is logical, the definition given is for its true state.

PARAMETER	Type	units
NF	Integer	

The number of states in the filter model.

NS Integer

The number of states in the truth model.

M Integer

The number of measurements to be processed at update time. If M is 0, updates are not attempted.

NZF Integer

The number of non-zero elements in the NFxNF filter dynamics partial matrix F.

NZQ Integer

The number of non-zero elements in the NFxNF filter noise strength matrix Qf.

NXTJ Integer

The number of variables, not counting time, to be read from the external 'trajectory' data tape (TAPE3).

LXTJ Logical

If TRUE, 'trajectory' data is external, i.e., available to SOFE on TAPE3. Each TAPE3 record is binary and consists of time followed by NXTJ variables. A description of TAPE3 structure is given in Subsection 4.1.2. If FALSE, any trajectory data required to find F, H, $\underline{DZ}$, $\underline{X}$s, etc., must be generated during the simulation in user-supplied routines, e.g. in TRAJ.

TO Real seconds (1)

Initial time of each run. Note: the last character in parameter names TO, TMEASO and HO is a zero.

(1) Seconds are shown here and elsewhere primarily for concreteness. If the user wishes to scale his problem in minutes, hours, days, etc., he may do so without altering SOFE.

TF	Real	seconds

Final time of each run. The simulation will not run backwards, so TF must be greater than TO.

TMEASO	Real	seconds

Updates are prohibited before this time.

DTMEAS	Real	seconds

The time interval between measurements. DTMEAS and the next five DT.... quantities are referenced to zero seconds; e.g., if TO = 50. and DTPRNT = 6., the simulation will pause to print output at T = (50., 54., 60., 66.,...).

DTPRNT	Real	seconds

Print interval. See LPR.

DTCCPL	Real	seconds

Data storage interval for Calcomp plots. See LCC.

DTPRPL	Real	seconds

Data storage interval for printer plots. Sampling will occur at a max of only 101 times. More samples are unwarranted because of limited printer plot resolution. See LPP.

DTSTIX	Real	seconds

The time interval between calls to user-subroutine ESTIX. The user may wish to use these calls to sample error, to construct some statistic, to output something not controlled by a PRDATA parameter, to modify a data base, etc. TAPE8 is provided for output from ESTIX.

DTNOYS Real seconds

The time interval between calls to user-subroutine SNOYS. These calls are for the purpose of adding noise to the truth state.

LPR Logical

Master control for printing. If TRUE, formatted writes are made to TAPE6 at synchronous intervals governed by DTPRNT. All other parameters that begin with the letters 'LPR' also control printing and are logically 'ANDED' with LPR. TAPE6 is for all listable output and will contain error messages, summary data, printer plots, etc., in addition to the synchronous output governed by LPR. See IPRRUN.

LPRXS Logical

Prints truth state vector $\underline{X}s$ at the interval specified by DTPRNT.

LPRXF Logical

Prints filter state vector $\hat{\underline{X}}f$ at DTPRNT interval.

LPRDG Logical

Prints the square root of the diagonal elements of Pf at DTPRNT interval. These are the one-sigma values for the states in $\hat{\underline{X}}f$.

LPRLT Logical

Prints the symmetric covariance matrix Pf in a lower triangular display at DTPRNT interval.

LPRUD Logical

Allows printed output of states and covariance values at update time. The previous five parameters still govern what is printed.

LPRZR Logical

Prints measurement residuals and their standard deviations at update time. These are the residuals ZRES=$\tilde{D}$Zj constructed by the user in subroutine HRZ. The residual standard deviation is the square root of $Rf+H*Pf*H^T$.

LPRH Logical

Prints the vector H at update time.

LPRK Logical

Prints the Kalman gain K at update time.

LPRXTJ Logical

Prints interpolated external trajectory data at DTPRNT intervals if LXTJ is true.

LCC Logical

Master control for Calcomp plotting. If TRUE, LCC enables storage of data on TAPE4 (at TO, DTCCPL intervals, update times and at TF) for subsequent Calcomp plotting by SOFEPL, [5].

LPP Logical

Master control for printer plotting. If TRUE, LPP enables storage of data on TAPE7 (at TO, DTPRPL intervals, update times if LPPUP is true, and at TF) during the first run for generation of printer plots at problem completion. Up to 20 plots may be made, each containing a max of 101 time samples. Control of what gets plotted is through data cards described below.

LPPLD Logical

Lists the data to be plotted when printer plots are generated. Each (x,y) pair for each curve is listed.

LPPUP Logical

Allows output to the printer plot file (TAPE7) at update times if printer plots are generated. The counterpart for printed output is LPRUD.

ICONT Integer

Control for $\underline{X}_{so}$, $\hat{\underline{X}}_{fo}$, and P_{fo} input. If set to 1, this problem is the continuation of a previous problem and the above values are read from TAPE9. If other than 1, these values are read from TAPE5 (new run).

ISEED Integer

Seed value for random number generator. ISEED is used to initialize the random number generator and thereby assure that a random number sequence can be repeated.

IPASS Integer runs

Number of runs over [TO,TF] in the Monte Carlo simulation.

IPRRUN Integer runs

Number of runs for which printed output is desired. IPRRUN has no effect if LPR is false. If LPR is true, all synchronous printed output is disabled after run IPRRUN is complete.

IPGSIZ Integer lines

The number of lines to be printed on each page. On most printers, approximately 60 lines fill one page. If more are printed, some will fall on the fold between pages unless the printer has its own page size control. If IPGSIZ is not positive, page control is turned off in SOFE.

MODE Integer

Indicates type of integration. If 1, the step size is 'variable', i.e., H is adjusted automatically to maintain the integration error below its allowed value. If not 1, the step size is fixed at H0. Variable step integration is strongly recommended.

TOLER Real

The permissible integration error per unit step when variable step mode is used. This tolerance, which applies equally to all variables, is applied as a relative (absolute) criterion when a variable's magnitude is >1 (<1).

HMAX Real seconds

Maximum permissible step size. This quantity is intended to be a measure of the 'scale' of the problem. If the integrator never uses a step size larger than HMAX, it should never step over, and therefore miss completely, any fluctuations in the solution. In effect, HMAX tells the integrator approximately how fine a mesh is needed for a reasonable attempt at solving the numerical integration problem.

HMIN Real seconds

Minimum permissible step size. If, in order to handle severe dynamics, the integrator reduces its step size to HMIN without satisfying the specified error criterion, an integration failure has occurred. If this happens, an error message is printed and the simulation stops.

H0 Real seconds

Initial integration step size. See MODE.

A word of explanation and caution is needed here. SOFE performs events in the order prescribed by the six DTs. When events occur simultaneously, SOFE does them in reverse order to the DT list above: i.e. calls to SNOYS first, calls to ESTIX second, ... , measurement processing last. To illustrate, if both a print and a measurement were scheduled at T=50, the print would be done first. Now suppose DTPRNT=3.4, DTMEAS=10.2 and T is approaching 30.6. This appears to be a simultaneous event situation, but this time the measurement will be done first because its event time computes to 30.599... while the prints event time computes to 30.600... . A computed event time can be slightly in error (either low or high) whenever its DT cannot be exactly represented in a full computer word, i.e. is not a power of two. For example, neither 3.4 nor 10.2 are so representable but 3.375 and 10.25 are. If the user wants to avoid the random ordering of events that can occur when T is a common multiple of several DTs, he must choose each DT to be exactly representable in one computer word.

Table 4-3

PRDATA RANGE AND DEFAULTS

Parameter	Range	Default
NF	> 0	1
NS	> 0	1
M	≥ 0	0
NZF	≥ 0	0
NZQ	≥ 0	0
NXTJ	≥ 0	0
LXTJ	T or F	F
TO	≥ 0	0.0
TF	> TO	1.0
TMEASO	≥ 0	0.0
DTMEAS	> 0	1.E+09
DTPRNT	> 0	1.E+09
DTCCPL	> 0	1.E+09
DTPRPL	> 0	1.E+09
DTSTIX	> 0	1.E+09
DTNOYS	> 0	1.E+09
LPR	T or F	T
LPRXS	T or F	T
LPRXF	T or F	T
LPRDG	T or F	T
LPRLT	T or F	F
LPRUD	T or F	F
LPRZR	T or F	F
LPRH	T or F	F
LPRK	T or F	F
LPRXTJ	T or F	F
LCC	T or F	F
LPP	T or F	F
LPPLD	T or F	F
LPPUP	T or F	F
ICONT	1 or not 1	0
ISEED	Unlimited	77
IPASS	> 0	1
IPRRUN	> 0	1
IPGSIZ	≥ 0	0
MODE	1 or not 1	1
TOLER	> 0	1.E-4
HMAX	> HMIN	1.E+9
HMIN	> 0	1.E-4
HO	> 0	1.E-2

4.1.1.2 Printer Plot Specification

SOFE was designed to make up to 20 printer plots. Each plot is a time history of user-specified variables as they evolved in the first run. The data for making the specified plots are saved on TAPE7 during the first run and then plotted after the last run. Data sampling, which occurs at TO, at DTPRPL intervals, at update times (if LPPUP is true) and at TF, is discontinued after 101 samples since 101 points saturate the time axis resolution. Note that when LPPUP is true, each update produces three samples at ti (one each at ti^-, ti^+ and ti^{+c}) which can rapidly fill the 101 available sample slots. Printer plots are shown in Appendices A and B.

Clearly printer plots are meant to provide only a quick-look capability. They will not satisfy the need for ensemble-of-runs plots or statistical calculations. This need is met by SOFEPL [5] as discussed in 4.2.2. If a user wants to go to the trouble of writing more code, ensemble statistics can also be computed in user-supplied routines as illustrated in subroutine ESTIX of the satellite orbit problem, Section 5.2 and Appendix B.

Each printer plot is specified by a 4-tuple called the 'plot parameter set'. This 4-tuple specifies what to plot against time and how the plot data are to be scaled. To il-

lustrate, suppose (for some perverse reason) one wanted to plot the difference Xs(5) - $\hat{X}$f(4) surrounded by $\pm$SQRT(Pf(7)) with a scale factor of 100 on all curves; the plot parameter set would be (5,4,7,100.).

In general, a plot parameter set is in the form NXS, NXF, NPF, R where NXS is the index for the required $\underline{X}$s variable, etc., and R is the scale factor applied equally to every curve. Missing variables are indicated by a 0 in a set. For example, (5,0,0,0.3) plots 0.3*Xs(5) versus time. Table 4-4 shows which curves are drawn for the six allowable combinations of NXS, NXF, NPF, R.

Table 4-4

PRINTER PLOT PARAMETER SET DEFINITIONS

	Parameter Set	Curve(s) Drawn (R Factor Omitted)
1	NXS, 0 , 0 ,R	Xs(NXS)
2	NXS, 0 ,NPF,R	Xs(NXS),Xs(NXS)$\pm$SQRT(Pf(NPF))
3	0 ,NXF, 0 ,R	$\hat{X}$f(NXF)
4	0 ,NXF,NPF,R	$\hat{X}$f(NXF),$\hat{X}$f(NXF)$\pm$SQRT(Pf(NPF))
5	NXS,NXF,NPF,R	Xs(NXS)-$\hat{X}$f(NXF),$\pm$SQRT(Pf(NPF))
6	NXS,NXF, 0 ,R	Xs(NXS),$\hat{X}$f(NXF)

4.1.2 TAPE3 Input, External Trajectory

TAPE3 contains unformatted records that are used to supply external trajectory information. An example of TAPE3 use would be to supply exact whole-valued position, velocity and attitude for the simulation of a navigation system. TAPE3 contains a header (optional) followed by fixed-length

records in binary format. If the trajectory is either internal or not required, LXTJ is set FALSE and TAPE3 is not used.

SOFE begins each run by rewinding TAPE3 and calling TRAJO which is responsible for reading the TAPE3 header. Doing things this way allows the user to write any header he desires since he must also write TRAJO. The header should probably be echoed to TAPE6 listable output, but only on run one.

After TRAJO has read the user's header, SOFE will begin to access trajectory data. Each fixed-length trajectory record is now read using a FORTRAN unformatted read statement like the following:

```
READ (3) T,(DUMMY(I),I=1,NXTJ)
```

Each TAPE3 record must contain time followed by NXTJ variables. Recall that NXTJ is specified to SOFE in PRDATA. The NXTJ variables are those chosen by the user and placed on TAPE3 for his particular problem.

The TAPE3 read process occurs in subroutine SPAN. Given simulation time T, SPAN surrounds T with trajectory data from three consecutive TAPE3 times (T1,T2,T3) such that

$T1 \leq T < T2 < T3$. If this inequality cannot be satisfied or if SPAN runs into an unexpected end-of-file (EOF), SPAN halts the program and writes a diagnostic message.

After SPAN has acquired the correct data, subroutine INTERP interpolates that data using cubic splines. Interpolation occurs at points in [T1,T3] where the integrator needs derivative evaluations of $\underline{X}s$, $\hat{\underline{X}}f$ and Pf. Current interpolated data are made available to the user by passing them in the argument list of every user routine except USRIN. See Section 4.3 and Table 4-5.

The following are points to remember about the external trajectory capability:

- o TAPE3 records need not be equally spaced in time but they must be in (ascending) chronological order.

- o The time spacing of TAPE3 data should be close enough to portray the activity in the trajectory. Shannon's sampling theorem applies here. However, oversampling could be costly in computer time because SOFE uses every TAPE3 record it sees.

- o T0 must not be less than the first TAPE3 time.

- o The next to last TAPE3 time must be strictly greater than TF.

These last two conditions may be summarized as follows:

$$\min T1 \leq T0 < TF < \max T2 \qquad (4\text{-}1)$$

4.1.3 TAPE9 Input, Previous Problem Continuing

When the present problem is the continuation of a previous problem, TAPE9 is required to supply $\underline{X}so$, $\hat{\underline{X}}fo$ and Pfo data to restart each run, IRUN=1,2,..., IPASS. If this is a brand new problem, not related to any previous problem, ICONT should be set to a value other than 1 to show TAPE9 is not required.

When TAPE9 is required, it must contain the following sequence of unformatted variable-length records:

Record	Length	Use
$\underline{X}so$	NS	
$\overline{X}fo$	NF	Used to continue run 1
$\overline{P}fo$	NTR	
$\underline{X}so$	NS	
$\overline{X}fo$	NF	Used to continue run 2
$\overline{P}fo$	NTR	
.	.	.
.	.	.
.	.	.
$\underline{X}so$	NS	
$\overline{X}fo$	NF	Used to continue run IPASS
$\overline{P}fo$	NTR	

TAPE9 will be configured this way automatically if it is the TAPE10 snapshot output from a previous run. SOFE will read TAPE9 in subroutines GETX and GETPF using the READ (9) ... statement.

4.2 Output

This section has four parts, one for each of the four tapes (numbered 4, 6, 8 and 10) that SOFE can output. The records on these tapes contain periodic summaries of the state of the truth and filter models. The output rate to each tape (except TAPE10) can be set independently by using control parameters in the PRDATA group.

4.2.1 TAPE6 Output, Listable Information

Listable output records consist of the following in the given order. Appendices A, B and C provide examples of listable output.

a. Banner, UPDATE Summary, Date and Time - The CDC banner page appears first followed by a page summarizing the subroutines and corrections in the SOFE deck. This page is generated by the UPDATE utility. At the bottom of this page are the date and time of the run.

b. SOFE Header Page - Contains general words naming the program, echoing the user's title, and again giving the run date and time.

c. Namelist and Nonzero Elements - Contains the PRDATA list of 40 parameters and two statements to echo the row-column pairs of the nonzero elements of F and Qf

d. Unlabeled Common - The next page of output begins with a table showing the structure of blank common area 'A'. A statement comparing the present size of A to its required size appears after the table. If A is too small, SOFE prints a message to this effect, tells the user to increase the size of A by altering two statements in the main program, and STOP's. A's present size is 1000.

e. User Input - Anything written by USRIN or by the entry phase of the other eight user-written routines

is output next.

f. Run Initialization Header - Each listed run begins with the user's title followed by run date, run time and run number.

g. Initial Values Echoed - $\underline{X}so$, $\hat{\underline{X}}fo$, SQRT[Pfo(i,i)] and Pfo are displayed at TO before problem execution begins. FALSE settings for LPR or for LPRXS, LPRXF, LPRDG and LPRLT will suppress all or any of these outputs.

h. Periodic Solution Printout - At DTPRNT intervals and at filter update times solution records are listed. The contents of these records are governed by the ten print control parameters that begin LPR. At DTPRNT intervals the maximum listed output is $\underline{X}s$, $\hat{\underline{X}}f$, the sigma value for each filter state, Pf shown in a lower triangular display and the interpolated trajectory data. At update times, if LPRUD is true, these same quantities are displayed just before the update (T-), $\hat{\underline{X}}f+$ and Pf+ are displayed after the update (T+), and $\underline{X}s+C$ and $\hat{\underline{X}}f+C$ are displayed just after the feedback correction (T+C). In addition, the measurement residuals, the standard deviations of those residuals, the H matrix computed by HRZ, and the Kalman gain may be displayed at update time. All output is in units specified by the user for his problem, e.g., in meters, seconds and radians. Units conversion is not done by SOFE.

i. Final Values - Same output set as at TO is printed at TF.

j. Printer Plots - Up to 20 plots are generated as specified by the plot parameter sets. If LPPLD is TRUE, the point pairs for each plot are also listed.

k. Final Check Product - This scaler is formed by chaining together in a single product all the nonzero elements of $\underline{X}s$, $\hat{\underline{X}}f$ and Pf at TF. This product is a handy index for checking whether a particular simulation run changed unexpectedly.

The repeatable output consists of items f. through i. which recur on every simulation run until the run number exceeds IPRRUN. The only output for runs after IPRRUN is a simple statement that the run was completed. All f.

through i. output is commanded to TAPE6 from the output executive routine 'OUT'.

Subroutine PAGCON maintains page control of listable output. All periodic print commands are preceded by a call to PAGCON indicating 1) the number of lines to print and 2) whether an a priori page eject should occur. When required, PAGCON issues a page eject of its own, thus maintaining line count and avoiding the breakup of output over a page boundary. When a printer has page size controls built in, one can save some computation time by setting IPGSIZ to zero, thereby disabling all calls to PAGCON.

4.2.1.1 Diagnostic Output

In addition to the planned-for output cited above, a diagnostic message is listed when SOFE detects a processing error. Tabulated below are the subroutines that produce such messages, together with the errors they detect. All these errors are fatal to further execution, so SOFE is STOPed if one occurs.

ADVANS:	Integration failed in KUTMER
GETPF:	IROW or ICOL > NF on input
INTERP:	Spline construction failed
INTERP:	Spline evaluation failed
PLPR:	More than 20 printer plots requested

PLPR:	Insufficient data cards
PSQRT:	Covariance not positive semidefinite
SOFEIN:	A dimension or size is out-of-range
SPAN:	TAPE3 data not in sync
SPLITA:	Blank COMMON too small
VALDTA:	An input parameter is out-of-range

The printed error message usually contains some information to help pinpoint the problem. In addition, any routine that reads input checks each FORTRAN READ for an EOF and STOP's the program when any EOF is found. The input subroutines are GETPF, GETX, NZRCIO, SOFEIN and SPAN.

About half of the above-listed processing errors arise from very specific flaws that are easily fixed, e.g. increasing the size of blank COMMON array A (and the DATA statement for MAXA) fixes the problem detected in SPLITA. Note that the dimension of array A must be no smaller than this:

4NF**2 + 15NF + 7NS + 2M + 3(NZF+NZQ) + 13NXTJ + 3

The remainder of the error messages arise for a variety of reasons that are not easily characterized. A case in point is an integration failure. This error may occur due to an inappropriate choice of the pair TOLER and HMIN, or to an

incorrect specification of derivatives in XFDOT or XSDOT, such as the insertion of a step change in some rate. This writer has not seen the errors cited in INTERP and SPAN occur in practice; the tests remain in place for safety. An indefinite covariance occurred once and was detected by PSQRT; it was caused by some incorrect off-diagonal terms in PF input.

4.2.2 <u>TAPE4</u> <u>Output, Calcomp</u>

When LCC is true, TAPE4 is generated to be the input to SOFEPL [5], the postprocessor program for the plotting of averages formed across an ensemble of Monte Carlo runs. TAPE4 consists of variable length records containing time, $\underline{X}s$, $\hat{\underline{X}}f$, the diagonal elements of Pf, measurement residuals and residual variances. These records are written to TAPE4 using unformatted binary writes. Data sampling, which occurs at T0, DTPLCC intervals, update times (ti^-, ti^+ and ti^{+c}) and TF, continues for all IPASS runs.

Using the data on TAPE4, SOFEPL can make 16 different types of plots (all versus time) with options available for scaling, time windowing and printing. To illustrate, one can plot $\overline{X}s(3)$, $\overline{\hat{X}}f(6)$, $\overline{E} = \overline{Xs(4)-\hat{X}f(3)}$, SE (which is the standard deviation of E), $\overline{Pf(3,3)**0.5}$, $\overline{ZRES(2)}$, etc., where the overbar indicates an ensemble average has been formed across a collection of Monte Carlo runs specified by the

user. SOFEPL uses the computer graphics language DISSPLA to generate an intermediate file that may later be linked to a Calcomp plotter, a CRT plotting terminal, or other graphics devices. Some plots made with SOFEPL are shown in Appendix B.

4.2.3 TAPE8 Output, User-Defined

TAPE8 is provided for user-defined output. Examples of such output might be error differences, normalized error quantities, data to interface SOFE with another processor, etc. A convenient place from which to write such output is user-written subroutine ESTIX which is called at DTSTIX intervals. Basic SOFE does nothing with TAPE8.

4.2.4 TAPE10 Output, Final Snapshot

TAPE10 is generated automatically by SOFE. At problem completion it contains a complete set of $\underline{X}$s(TF), $\hat{\underline{X}}$f(TF) and Pf(TF) values for each run. TAPE10 is closed with an EOF mark after the IPASSth run is complete. The use of this data for problem continuation is discussed in 4.1.3.

4.3 User-Written Subroutines

Basic SOFE consists of 31 routines that are constant from one problem to the next. SOFE 'adapts' to new problems by accepting user-written subroutines that define the variable aspects of every new application. This section outlines the purpose and minimum requirements for each user-written subroutine. We list their names below:

AMEND	ESTIX	FQGEN
HRZ	SNOYS	TRAJO
USRIN	XFDOT	XSDOT

Each user routine is a FORTRAN subroutine. Except for USRIN and TRAJO, each must contain a FORTRAN ENTRY statement. The entry name is formed by adding an '0' to the subroutine's name, e.g., AMEND0 from AMEND. These entry names are called once at the beginning of each run in order to initialize the data or variables particular to that routine. The run number (1,2,...,IPASS) is supplied in the argument list to allow the user to modify or inhibit this initialization as he might desire from run to run.

Definitions for the parameters and variables that appear in the argument lists of the user routines are given in Table 4-5. The first twelve quantities in Table 4-5 are for input and must not be altered in any user routines. The routines that may output and thereby alter the last eight

Table 4-5

DEFINITION OF QUANTITIES IN USER-ROUTINE ARGUMENT LISTS

FORTRAN Name	Quantity Definition	Math Symbol	Defining Equation
IRUN	Run number	-	-
T	Current simulation time	t	2-1 or 2-4
NF	Number of filter states	NF	2-4
NS	Number of truth states	NS	2-1
NTR	Size of triangular part of Pf	NTR	2-38
M	Number of measurements	M	2-2
NZF	Number of nonzero elements in F	-	-
NZQ	Number of nonzero elements in Qf	-	-
NXTJ	Number of variables read from TAPE3	-	-
IMEAS	Index of current measurement (1 thru M)	J	e.g. 2-33
XTRAJ	External trajectory data interpolated at T	-	-
PF	Filter covariance (upper triangular storage)	Pf(t)	2-8
XF	Filter state vector estimate	$\hat{X}f(t)$	2-4
XS	Truth state vector	Xs(t)	2-1
H	Measurement sensitivity vector, namely $H_J(1), \ldots, H_J(NF)$	H_J	2-33
RF	Measurement noise variance, a scalar specifying Rf_J	Rf_J	2-34
ZRES	Measurement residual, a scalar that simulates the difference between the actual Jth measurement and its predicted value	$\tilde{D}Z_J$	2-35
F	Filter dynamics partial matrix, nonzeros only	$F(t;\hat{X}f)$	2-16
QF	Filter driving noise strength, nonzeros only	Qf(t)	2-4
XDOT	Derivatives of Xs and $\hat{X}f$	$\dot{X}s$ & $\dot{\hat{X}}f$	2-1 and 2-27

quantities in Table 4-5 are mentioned in the individual descriptions that follow next. Examples of each routine are given in Appendices A and B.

PF,XF and XS are FORTRAN names for Pf, $\hat{\underline{X}}$f and $\underline{X}$s. H, RF and ZRES are FORTRAN names for Hj, Rfj and $\tilde{D}$Zj as defined by equations (2-33), (2-34) and (2-35) respectively. F and QF are FORTRAN names for F(t;$\hat{\underline{X}}$f) and Qf(t) as defined by (2-16) and the remarks after (2-4) respectively. XDOT is the homogeneous part of $\dot{\underline{X}}$s or $\dot{\hat{\underline{X}}}$f; it is computed as $\underline{g}$($\underline{X}$s,t) in subroutine XSDOT and as $\underline{f}$($\hat{\underline{X}}$f,t) in subroutine XFDOT. Naturally, the user is free to alter any of these FORTRAN names to better suit his problem or his preferences. The FORTRAN names in Table 4-5 will be used in the user-routine descriptions that follow now and also in coding for the two examples in Section 5.

4.3.1 AMEND

Subroutine AMEND is called at ti (after all M measurements have been processed) to apply feedback of newly computed estimates to both filter and truth states. A typical use of this routine would be to implement total, impulsive control in which:

- o All filter estimates are zeroed: XF,new = 0.
- o The corresponding truth states are bumped by the same amount as the filter states changed.

If the user wished to implement an open loop system, he would do nothing to alter either XS or XF in AMEND. Other feedback schemes, including continuous control, can be achieved by using AMEND in conjunction with the derivative routines XSDOT and XFDOT. Table 4-6 shows the layout of AMEND with the required statements in capital letters.

Table 4-6

REQUIRED STATEMENTS FOR AMEND

```
SUBROUTINE AMEND(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)
feedback computations
RETURN
ENTRY AMENDO
initialization
RETURN
END
```

4.3.2 ESTIX

Subroutine ESTIX is called at DTSTIX intervals to compute whatever quantities the user desires. TAPE8 is provided for storing these quantities if so required. Table 4-7 shows the layout of ESTIX with the required statements in capital letters.

Table 4-7

REQUIRED STATEMENTS FOR ESTIX

```
 SUBROUTINE ESTIX(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,
*                 NTR,PF)
 DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR)
 computations for user quantities
 RETURN
 ENTRY ESTIXO
```

```
initialization
RETURN
END
```

4.3.3 FQGEN

Subroutine FQGEN computes the values for the nonzero elements of the filter matrices F and QF where F is defined by equation (2-16) and QF by equation (2-4). F and QF are used in FPPPFT and ASYSP to compute the derivative

$$\dot{PF} = F*PF + PF*F^T + QF \tag{2-28}$$

It is important to note that the indices assigned to the nonzero values of F and QF in FQGEN must agree with the order in which the nonzero row-column pairs were specified on input. That is, using F for illustration, the first nonzero row-column pair is associated with F(1), the second with F(2), etc. Any order is allowed so long as the nonzero row-column pair order and the F element index order agree.

Table 4-8 shows the layout of FQGEN with the required statements in capital letters. Note that the ENTRY area is a convenient and efficient place for assigning the time invariant values of F and QF. Also note that FQGEN is in the innermost integration loop which means it is called, along with XFDOT and XSDOT, much more frequently than most other routines. The user should therefore endeavor to write effi-

cient code in constructing FQGEN, XFDOT and XSDOT.

Table 4-8

REQUIRED STATEMENTS FOR FQGEN

```
 SUBROUTINE FQGEN(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,
*              NZF,NZQ,F,QF)
 DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),F(NZF),QF(NZQ)
 computations for nonzeroes of F  and  QF  in  proper
 order
 RETURN
 ENTRY FQGENO
 initialization
 RETURN
 END
```

4.3.4 HRZ

Subroutine HRZ is called M times at each measurement update time to compute values for the vector H and the scalars RF and ZRES. The user schedules measurement updates at desired times by proper choice of the input parameters TMEASO and DTMEAS. He can suppress a particular measurement at any update time by returning a negative RF which SOFE interprets as a command to increment IMEAS and then proceed immediately to the next measurement. In short, every update session results in M calls from SOFE to HRZ, with the update algebra invoked in SOFE after any call in which RF is nonnegative. A varied measurement schedule can be arranged by the appropriate combination of input parameter choices and logic in HRZ.

Function subprogram GAUSS(AMEAN,STD) is available for computing random samples from a Gaussian distribution to aid in constructing ZRES. AMEAN and STD are the mean and standard deviation of the desired distribution. Table 4-9 shows the layout of HRZ with the required statements in capital letters.

Note that PF, one of the formal parameters in the HRZ argument list, is not used in either the linear or the extended Kalman filter. It is there because most higher-order filter mechanizations, e.g. the Gaussian second-order filter, require it to form a bias term for compensating ZRES. PF is also included in the argument list of XFDOT in anticipation of higher-order filter needs.

Table 4-9

REQUIRED STATEMENTS FOR HRZ

```
      SUBROUTINE HRZ(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF,
     *               IMEAS,M,H,RF,ZRES)
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),H(NF)
      computations for H,RF and ZRES
      RETURN
      ENTRY HRZO
      initialization
      RETURN
      END
```

4.3.5 SNOYS

Subroutine SNOYS is called at DTNOYS intervals to allow the user to inject noise into the appropriate states of XS.

These are the truth states influenced by the $\underline{w}s(t)$ term in equation (2-1). The usual procedure here is: first, obtain a delta covariance based on Qs that accounts for the random portion of the growth of the covariance of the XS process over the previous DTNOYS seconds; second, using this delta covariance, sample from a Gaussian distribution to obtain a random sample $\underline{w}d$; and third, add $\underline{w}d$ to XS to produce a perturbed truth state. The sampling procedure can use function GAUSS, which was described in HRZ previously. Table 4-10 shows the layout of SNOYS with the required statements in capital letters.

Table 4-10

REQUIRED STATEMENTS FOR SNOYS

```
SUBROUTINE SNOYS(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)
computations to inject noise in XS
RETURN
ENTRY SNOYSO
initialization
RETURN
END
```

4.3.6 TRAJO

The phrase 'trajectory data' comes out of the INS context where it usually means true, whole-valued position, velocity and attitude. Such data can be produced during SOFE simulation runs (e.g. by inclusion in XS) or drawn in from an external trajectory tape. The latter approach is generally preferred because it reduces the computational

load; SOFE is fully ready to accommodate this approach (see Subsection 4.1.2 on TAPE3 input).

However, when trajectories need to be generated during SOFE runs, TRAJ could be constructed and called by the user to do the job. SOFE itself never calls TRAJ. Therefore, unless and until the user assigns TRAJ a job to do, it is not needed.

TRAJO, on the other hand, is called by SOFE at the beginning of each new run to initialize TRAJ if necessary or to read the TAPE3 header if LXTJ is TRUE. Table 4-11 shows a layout for both TRAJ and TRAJO (although only TRAJO is essential) with the required statements in capital letters.

Table 4-11

REQUIRED STATEMENTS FOR TRAJO

```
SUBROUTINE TRAJ(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)
COMMON/.../...(optional)
computation of actual trajectory quantities (optional)
RETURN
ENTRY TRAJO
initialize actual trajectory (optional)
read TAPE3 header (required if LXTJ = .T.)
RETURN
END
```

4.3.7 USRIN

Subroutine USRIN is called once by SOFE during problem initialization for the primary purpose of reading

AD-A093 887 AIR FORCE WRIGHT AERONAUTICAL LABS WRIGHT-PATTERSON AFB OH F/G 9/2
SOFE: A GENERALIZED DIGITAL SIMULATION FOR OPTIMAL FILTER EVALU--ETC(U)
OCT 80 S H MUSICK
UNCLASSIFIED AFWAL-TR-80-1108 NL

2 OF 3
AD
A093887

END
DATE
FILMED
2-81
DTIC

CONT.

user-supplied data. However, USRIN could be used to perform any function that needed to be done only once at problem startup. If card input data are to be read by USRIN, those data must be inserted between the initial covariance data and the printer plot data (see Table 4-2). Table 4-12 shows the layout of USRIN with the required statements in capital letters. Note that USRIN has no ENTRY statement.

Table 4-12

REQUIRED STATEMENTS FOR USRIN

```
SUBROUTINE USRIN
read user data
RETURN
END
```

4.3.8 XFDOT

Subroutine XFDOT computes the homogeneous part of the derivative XDOT of the filter state vector XF. This derivative is given by (2-27) as

$$XDOT = \underline{f}(XF,t) \qquad (2\text{-}27)$$

XDOT output is used to propagate the filter state vector between updates via numerical integration. As mentioned before in 4.3.3, this routine is called often, so efficiency of coding here could significantly shorten run time.

We remark that (2-27) is the most general equation, descriptive of the nonlinear situation when the extended Kalman filter is appropriate. If (2-27) can be reorganized as

$$XDOT = F(t)*XF \tag{4-2}$$

linear filter principles apply and the user's job is simplified. To wit:

- o The nonzero values of F(t) computed for (4-2) are the ones required for F in FQGEN. Sharing of these values through a user-defined labeled COMMON area should result in computer savings. Note that XFDOT is called before FQGEN.

Note that basic SOFE makes no distinction between linear and nonlinear problems. All such differences are manifested in the user's computations for $\underline{f}$, F, $\underline{h}$, H and ZRES. Table 4-13 shows the layout of XFDOT with the required statements in capital letters.

Table 4-13

REQUIRED STATEMENTS FOR XFDOT

```
 SUBROUTINE XFDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,
*                NTR,PF,XDOT)
 DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),XDOT(NF)
 computations to form XDOT
 RETURN
 ENTRY XFDOTO
 initialization
 RETURN
 END
```

4.3.9 XSDOT

Subroutine XSDOT computes the homogeneous part of the derivative of the truth state vector XS. This derivative, which again carries the FORTRAN name XDOT, is given from (2-1) as

$$XDOT = \underline{g}\ (XS,t) \tag{4-3}$$

XDOT is used to propagate the truth state via numerical integration between noise addition points. As with FQGEN and XFDOT, XSDOT is called often, so efficiency in coding is important. Table 4-14 shows the layout of XSDOT with the required statements in capital letters.

Table 4-14

REQUIRED STATEMENTS FOR XSDOT

```
SUBROUTINE XSDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,XDOT)
DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),XDOT(NS)
computations to form XDOT
RETURN
ENTRY XSDOTO
initialization
RETURN
END
```

4.3.10 Summary of User-Written Routines

The following is a short summary of Subsection 4.3. It provides a brief definition of what calculations each user-written routine must perform and gives the appropriate equations as required.

o**AMEND**

:Apply Feedback

o**ESTIX**

:User-Defined Output

o**FQGEN**

:Calculate Nonzero Values of F and QF

$$F(t) = \left. \frac{\partial \underline{f}(\underline{X}f,t)}{\partial \underline{X}f} \right|_{\underline{X}f = \hat{\underline{X}}f}$$

$$E[\underline{w}f(t)\underline{w}f^T(t+T)] = Qf(t) * \delta(T)$$

o**HRZ**

: Calculate H, Assign RF, Simulate ZRES

$$H(ti) = \left. \frac{\partial \underline{h}f(\underline{X}f,ti)}{\partial \underline{X}f} \right|_{\underline{X}f = \hat{\underline{X}}f}$$

$$E[\underline{v}f(ti)\underline{v}f^T(tj)] = Rf(t)*\delta ij$$

$$ZRES = \begin{cases} \underline{h}s(\underline{X}s,ti) + \underline{v}s - \underline{h}f(\hat{\underline{X}}f,ti) & \text{nonlinear} \\ Hs*\underline{X}s + \underline{v}s - H*\hat{\underline{X}}f & \text{linear} \end{cases}$$

o**SNOYS**

: Inject Additive Noise $\underline{w}d$ Into Truth States

$$E[\underline{w}d(tj)\underline{w}d^T(tj)] = Qs(tj) * \Delta t$$

o$\underline{\text{TRAJ}}$

: Supply Trajectory Data for Computing

$F, Qf, \dot{\underline{X}}s, \dot{\hat{\underline{X}}}f, \underline{h}s, \underline{h}f$, etc.

o$\underline{\text{USRIN}}$

: User-Defined Input

o$\underline{\text{XFDOT}}$

: Calculate Filter Derivatives (Homogeneous Part Only)

$$\dot{\hat{\underline{X}}}f = \begin{cases} \underline{f}(\hat{\underline{X}}f, t) & \text{nonlinear} \\ F(t) * \hat{\underline{X}}f & \text{linear} \end{cases}$$

o$\underline{\text{XSDOT}}$

: Calculate Truth Derivatives (Homogeneous Part Only)

$$\dot{\underline{X}}s = \underline{g}(\underline{X}s, t)$$

$$\dot{\underline{X}}s = \begin{cases} \underline{g}(Xs, t) & \text{nonlinear} \\ G(t) * \underline{X}s & \text{linear} \end{cases}$$

5.0 EXAMPLE PROBLEMS

This section presents examples of SOFE and SOFEPL use for two simple problems. The first example, taken from [1] and [2], is for a totally linear system. The second example is a nonlinear orbit determination problem that is studied using extended Kalman filter techniques. For each example we describe the truth model and the reduced-order filter model, summarize the example's implementation in SOFE, and discuss the results.

5.1 Linear System Example

This linear example is a simplified, undamped, single-axis inertial navigation system (INS) having position and velocity measurements for outputs. In general, the navigation equations for an INS are nonlinear. By confining our attention to the error states of this simplified INS, we construct a purely linear example.

5.1.1 Truth Model

The model for the truth system is shown in Figure 5-1. The truth system consists of a single-axis INS driven by gyro and accelerometer sensor errors. In the figure, R is earth's radius and g is acceleration due to gravity. The system outputs are biased measurements of position and velocity that occur every 30 seconds.

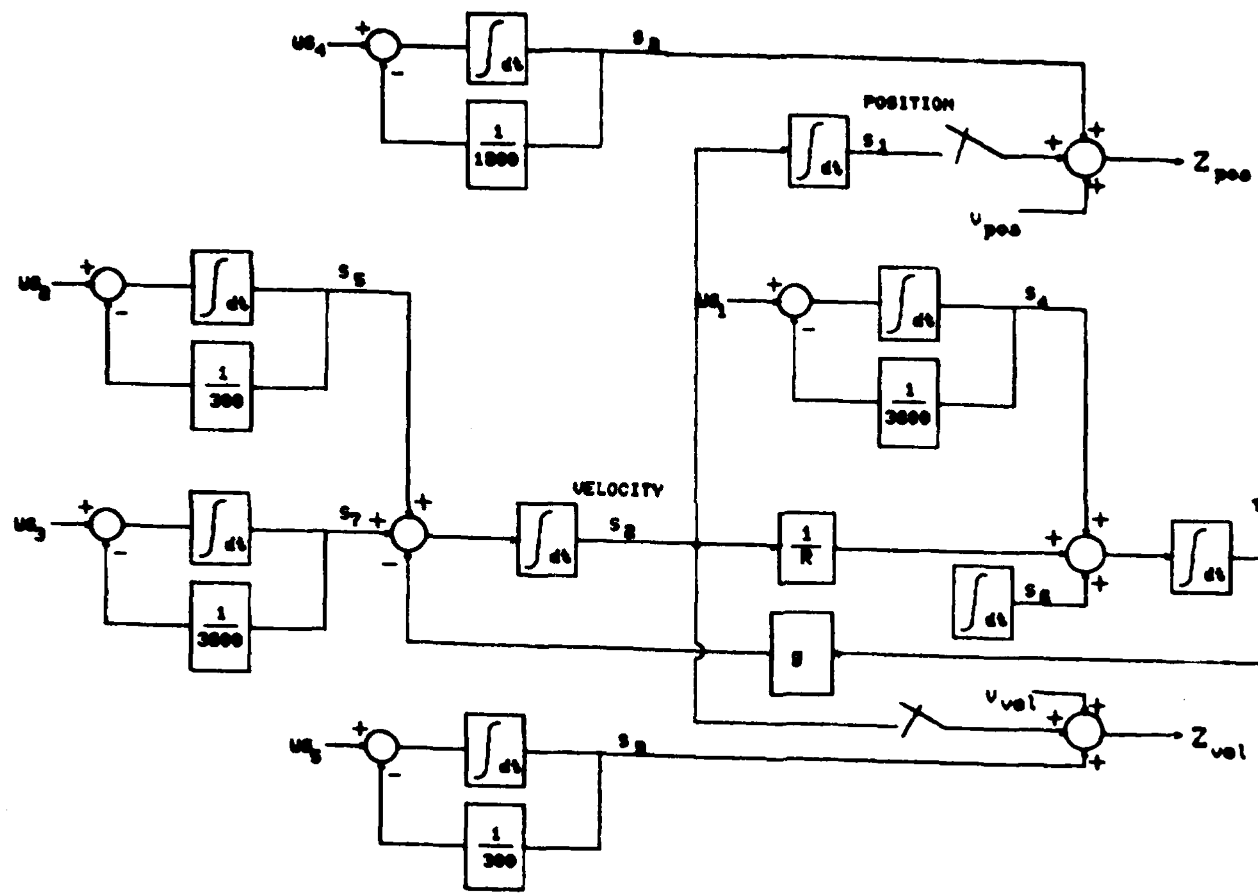


Figure 5-1. INS Truth Model Block Diagram.

The basic INS truth system consists of position error (s1), velocity error (s2), and tilt error (s3). Tilt is driven by two random drift processes; the first is a first-order Markov process (s4) while the second is a random constant (s6). Velocity error is driven by two random acceleration processes, both first-order Markov, one having a long correlation time (s7) and one a short correlation time (s5). The other two states in the truth model are biases on the measurements. Both measurement biases are first-order Markov processes with s8 being the bias on position and s9 the bias on velocity. The above information is summarized in Table 5-1 where data on the nature and statistics of each random process are also given.

Table 5-1

DEFINITION OF TRUTH MODEL STATES

State	Description	Process Type	Sigma Value	Correlation Time
s1	Position error	-	-	-
s2	Velocity error	-	-	-
s3	Tilt error	-	-	-
s4	Drift bias	1st Markov	.01 deg/hr	3600
s5	Accel. bias	1st Markov	200E-6 g's	300
s6	Drift bias	Random constant	.005 deg/hr	infinite
s7	Accel. bias	1st Markov	100E-6 g's	3600
s8	Pos.meas.bias	1st Markov	300 ft.	1800
s9	Vel.meas.bias	1st Markov	0.5 ft/sec	300

Denoting $\underline{X}s$ as the set of states (s1,s2,...,s9), the differential equation for the truth system is linear and may be written in state-space form as:

$$\dot{\underline{X}}s = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1/R & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/3600 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/300 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/3600 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/1800 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/300 \end{bmatrix} \underline{X}s + \begin{bmatrix} 0 \\ 0 \\ 0 \\ ws1 \\ ws2 \\ 0 \\ ws3 \\ ws4 \\ ws5 \end{bmatrix} \tag{5-1}$$

The vector output equation is also linear in $\underline{X}s$

$$\underline{Z}s = \begin{bmatrix} Zpos \\ Zvel \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \underline{X}s + \underline{v}s \tag{5-2}$$

The two measurements are uncorrelated and have covariance

$$Rs = \begin{bmatrix} 100.**2 & 0 \\ 0 & 0.5**2 \end{bmatrix} \tag{5-3}$$

where position and velocity units are feet and ft/sec. Note that ( )**2 means ( ) squared.

5.1.2 Filter Model

One primary concept for a filter model is that it

should be computationally simple. This is usually accomplished in part by deleting states from the truth model that are deemed non-significant. In the case at hand, the ast four states of $\underline{X}s$, viz. s6 through s9, were dropped, leaving a dynamic model for $\underline{X}f$ like that for truth states s1 through s5. Compensation for this mismodeling would typically include increasing the noise at the ports where the deleted states drove the system and increasing the measurement error variance. Neither compensation will be used in this implementation so we may obtain results comparable to those of [1] and [2].

The filter model equations corresponding to (5-1) and (5-2) are therefore

$$\dot{\underline{X}}f = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 1 \\ 0 & 1/R & 0 & 1 & 0 \\ 0 & 0 & 0 & -1/3600 & 0 \\ 0 & 0 & 0 & 0 & -1/300 \end{bmatrix} \underline{X}f + \begin{bmatrix} 0 \\ 0 \\ 0 \\ wf1 \\ wf2 \end{bmatrix} \tag{5-4}$$

$$\underline{Z}f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \underline{X}f + \underline{v}f \tag{5-5}$$

The strengths of the noises wf1 and wf2 were chosen to equal those of ws1 and ws2. Similarly, the standard deviations of the position and velocity measurement noises were left unchanged so Rf is

$$Rf = \begin{bmatrix} 100.**2 & 0 \\ 0 & 0.5**2 \end{bmatrix} \tag{5-6}$$

The initial covariance matrix is diagonal and has the values listed in Table 5-2. Smaller initial values were used here than in [1] and [2] in order to narrow the dynamic range of the covariance output so the printer plots will better show what happens.

Table 5-2

INITIAL FILTER COVARIANCE - INS PROBLEM

Filter State,i	Pfii(0) English Units		Computation Units	
1	(120	ft)**2	14400	ft**2
2	(2	fps)**2	4	(fps)**2
3	(0.1	deg)**2	3.046E-6	rad**2
4	(0.01	deg/hr)**2	2.350E-15	(rad/sec)**2
5	(200	micro g's)**2	4.147E-5	(fps2)**2

5.1.3 SOFE Implementation

The subroutines required to implement the foregoing problem are shown at the beginning of Appendix A. The input data for this problem were previously shown in Figure 4-2. Some notes about this implementation follow:

- o Total impulsive control was implemented in AMEND.

- o Since F and Qf are constant, all computations for these quantities were done in FQGENO.

o The INS was at rest on the earth's surface, so trajectory computations were not needed, either internally or externally. Thus TRAJ is eliminated and TRAJO is little more than a program stub.

o Given a first order Markov process with time constant TAU and steady state variance SIGMA**2, the increase in process variance over an interval DT is (SIGMA**2) * (1-EXP(2*DT/TAU)) which is approximated for small DT/TAU (in SNOYS) by (SIGMA**2) * (2*DT/TAU)

o USRIN was used to read the statistical data describing the Markov processes and the measurement noises. Two NAMELISTS were set up for this purpose, one for the filter (INF) and one for the truth (INS).

o Function subroutine GAUSS, supplied for the user's convenience in basic SOFE, was used in HRZ, in SNOYS, and in XSDOT to obtain random Gaussian samples. GAUSS has two arguments, the first being the sample's desired mean and the second its standard deviation. GAUSS was used in HRZ to simulate the measurement noise vf, in SNOYS to simulate the change in Xs due to driving noise ws, and in XSDOT to initialize the random constant state s6.

o The problem was set up as a single run (IPASS=1) of ten hours duration.

5.1.4 Results and Conclusions

Appendix A contains the printed output for this sample problem. Note that:

o The data echoed on the early pages of the printout matches that prescribed by Figure 4-2.

o Printout is disabled at update times - 1200 updates occurred - to avoid excessive output. Periodic output occurs only at one hour intervals.

o A total of five plots have been made. The plot titles have been chosen to match their plot parameter set input (see Table 4-4).

o The last page of printed output is the dayfile showing the CDC job control used in making this run. Note the parameters on the first card, the job card, and the time required to complete the run. Note also that the source code, including the user programs, is being carried in CDC "UPDATE" format. Both this code and the data are stored as permanent files on the CDC disc. The required information for accessing these files is given on the ATTACH cards.

The printer plots near the end of Appendix A show that filter states $\hat{X}f2$, $\hat{X}f3$, $\hat{X}f4$ and $\hat{X}f5$ track the corresponding truth states within acceptable covariance limits. This point might be argued in the case of state $\hat{X}f2$ since the true error is occasionally over 3 sigma from the error as estimated by Pf. Since state $\hat{X}f2$ always rights itself we have chosen to label its performance acceptable.

However, the first printer plot clearly shows that $\hat{X}f1$ does not track Xs1. Pf11 shrinks to less than 10 feet in the first hour (a "*" on the plot indicates coincident points, in this case between 2, which corresponds to +SQRT(Pf11) and 3, which corresponds to -SQRT(Pf11)) while the difference Xs1-$\hat{X}f1$ (represented by curve 1) meanders about aimlessly. Xs8, being a large unmodeled bias on each position measurement, is probably the root cause of divergence in $\hat{X}f1$.

Since one state is divergent, the filter is considered divergent and would require either redesign or tuning (adjustment of parameters) to get acceptable performance.

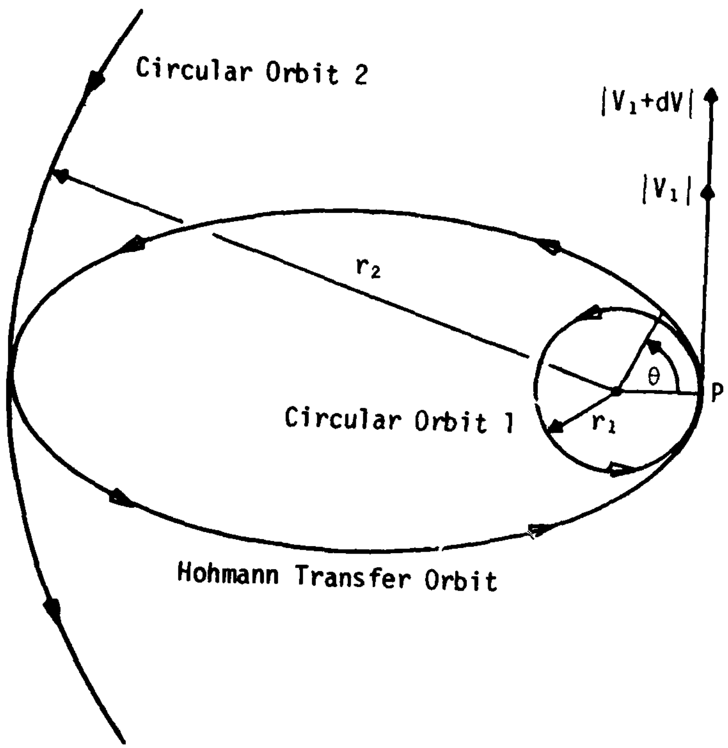
5.2 Nonlinear System Example

The following are the in-plane equations of motion for a unit mass in an inverse square law force field:

$$\ddot{r} = r\dot{\theta} - Go/r \tag{5-7}$$

$$\ddot{\theta} = -2\dot{r}\dot{\theta}/r \tag{5-8}$$

In (5-7) and (5-8), r is the range from the center of the force field to the unit mass (satellite) and θ is the angle between the unit mass and a reference line passing through the center of the field and fixed in space. For convenience, we let Go = 1 and r(TO)=1. In addition, if $\ddot{r}$(TO) is zero, the forces produced at TO are roughly 1/32 those on earth's surface. The general solution is an orbit along some conic section, two versions of which are a circle and an ellipse, Figure 5-2. When the ellipse is a transfer path from one circular orbit of radius r1 to another of radius r2 and the speed change required to achieve the transfer is dv, the doubly-tangent elliptical path that results is called the Hohmann transfer orbit. These orbits are determined by the initial conditions on (5-7) and (5-8).



r_1 = radius of circular orbit 1
= perigee of ellipse

r_2 = radius of circular orbit 2
= apogee of ellipse

$|V_1|$ = velocity at P for circular orbit 1

$|V_1+dV|$ = velocity at P for Hohmann transfer orbit

$$\left(\frac{dV}{V_1}\right)^2 = 3 - \frac{2r_1}{r_1+r_2} - 2\sqrt{\frac{2r_2}{r_1+r_2}}$$

Figure 5-2. Satellite Orbit Geometry

Given measurements of r and θ, the problem will be to determine the position (r,θ) and velocity $(\dot{r},\dot{\theta})$ of the satellite. This example is an adaptation of material from Reference 6.

5.2.1 Truth Model

Define the state vector $\underline{X}s$ as

$$\underline{X}s = \begin{bmatrix} Xs1 \\ Xs2 \\ Xs3 \\ Xs4 \end{bmatrix} = \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{5-9}$$

and rewrite (5-7) and (5-8) in state-space form as

$$\dot{\underline{X}}s = \underline{g}\ (Xs,t) = \begin{bmatrix} Xs2 \\ Xs1*Xs4**2 - Go/Xs1**2 \\ Xs4 \\ -2*Xs2*Xs4/Xs1 \end{bmatrix} \tag{5-10}$$

Note the peculiar absence of random driving terms in (5-10). Phenomena such as solar pressure, atmospheric drag, gravity anomalies, outgassing and satellite tumbling could add uncertainty to (5-10) but these factors have been ignored. The truth model thereby becomes an idealized representation of nature. One consequence is to make the implementation in SOFE somewhat easier: (5-10) provides the equations for XSDOT while SNOYS has no role at all. Ground observations of r and θ are available every 0.5 time units,

$$\underline{Zs}(ti) = \begin{bmatrix} Xs1(ti) + vs1(ti) \\ Xs3(ti) + vs3(ti) \end{bmatrix} \tag{5-11}$$

where $\underline{v}s$ is a zero-mean white Gaussian noise having a diagonal 2x2 covariance matrix Rs given by

$$Rs = \begin{bmatrix} 0.1{*}{*}2 & 0 \\ 0 & 0.2{*}{*}2 \end{bmatrix}$$

5.2.2 Filter Model

The highly nonlinear nature of the satellite equations of motion suggests very strongly that the filter propagation equations must also be nonlinear. Thus the following equations were adopted to model the satellite motion for the filter:

$$\dot{\underline{X}}f = \underline{f}(\underline{X}f,t) + \underline{w}f(t)$$

$$= \begin{bmatrix} Xf2 \\ Xf1{*}Xf4{*}{*}2-Go/Xf1{*}{*}2 \\ Xf4 \\ -2{*}Xf2{*}Xf4/Xf1 \end{bmatrix} + \begin{bmatrix} 0 \\ wf1 \\ 0 \\ wf2 \end{bmatrix} \tag{5-12}$$

The states in $\underline{X}f$ correspond one-for-one to those in $\underline{X}s$. The distinction between (5-12) and (5-10) is that the two random driving processes wf1 and wf2 have been added to account for orbit uncertainties.

The filter measurement equation is

$$\underline{Z}f(ti) = \begin{bmatrix} Xf1(ti) + vf1(ti) \\ Xf3(ti) + vf3(ti) \end{bmatrix} \tag{5-13}$$

where $\underline{v}f$ is a zero-mean white Gaussian noise having a diagonal 2x2 covariance matrix Rf given by

$$Rf = \begin{bmatrix} 0.1**2 & 0 \\ 0 & 0.2**2 \end{bmatrix}$$

Since the filter model is nonlinear, the extended Kalman filter will be used. The state propagation equation is

$$\dot{\hat{\underline{X}}}f = \underline{f}(\hat{\underline{X}}f,t) \tag{5-14}$$

where $\underline{f}(.)$ is given by (5-12). In order to keep track of the covariance of the error $\underline{DX}$ in $\hat{\underline{X}}f$, $F(t;\underline{X}f)$ was established using (2-16):

$$F(t;\underline{X}f) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ Xf4**2+2Go/Xf1**3 & 0 & 0 & 2Xf1*Xf4 \\ 0 & 0 & 0 & 1 \\ 2Xf2*Xf4/Xf1**2 & -2Xf4/Xf1 & 0 & -2Xf2/Xf1 \end{bmatrix} \tag{5-15}$$

Also, based on (5-12), Qf is

$$Qf = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & Qf1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Qf2 \end{bmatrix} \tag{5-16}$$

The nonzero elements in (5-15) and (5-16) are used to form $\dot{P} = FP+PF^T+Qf$ to propagate the covariance.

The filter processes two measurement differences at update time:

$$\underline{\tilde{DZ}} = \begin{bmatrix} Xs1 + vs1 - \hat{X}f1 \\ Xs3 + vs3 - \hat{X}f3 \end{bmatrix} \tag{5-17}$$

By inspection of (5-17) H is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5-18}$$

The initial covariance matrix Pfo is diagonal with values listed in the table below.

Table 5-3

INITIAL FILTER COVARIANCE - ORBIT PROBLEM

Filter State, i	Pfo(i, i)	Units
1	0.1	length**2
2	0.1	(length/unit time)**2
3	0.1	radians**2
4	0.1	(radians/unit time)**2

5.2.3 SOFE Implementation

The input data for this study are shown in Appendix D.

The subroutines required to implement this nonlinear problem are included with the listed output in Appendix B. Some notes about this implementation follow:

o Since subroutine XPLUS in basic SOFE′ computes $\underline{\hat{X}}f^{+}$ as

$$\underline{\hat{X}}f^{+} = \underline{\hat{X}}f^{-} + K*ZRES \qquad (5\text{-}19)$$

the whole-valued estimate $\underline{\hat{X}}f$ is completely corrected at ti^{+}. No further corrective action is required so AMEND was merely a RETURN statement.

o Since Qf is constant, it was computed once for all time in FQGENO.

o Since the truth model was devoid of random driving noise, SNOYS was only a stub in the load module.

o As with the linear example, no trajectory computations were needed (outside those in XSDOT and XFDOT) so TRAJ is not required at all, and TRAJO is required only as a stub.

o Two different truth orbits were implemented in this study. The first truth orbit was circular while the second was elliptical with r2:r1 = 10:1. Since initial conditions completely determine orbit, the two orbits were created using the following values at TO:

(1. 0. 0. 1.) circular orbit

(1. 0. 0. 1.3483997245) elliptical orbit

The above values initialized both $\underline{X}s$ and $\underline{\hat{X}}f$. The circular amd elliptical orbit studies were conducted in separate computer runs. The output in Appendix B and the job control in Appendix D illustrate the circular orbit case.

- o Subroutine ESTIX was constructed to compute the average sigma (Pf(i,i)**0.5) as well as the mean and standard deviation of true error for an ensemble of Monte Carlo runs. True error in state $\hat{X}f(i)$ is the difference Xs(i) - $\hat{X}f(i)$. In this study, sampling of sigma and true error values occurred at DTSTIX = 0.5 time units, just <u>before</u> each update, on all four states. When T reached TF on the last run, ESTIX computed the appropriate statistics for each state at each sample time for the ensemble of IPASS runs. These ensemble statistics are printed just before the printer plots in Appendix B.

- o TF was 5 time units, Qf1 was 0.02 length2/time3, and Qf2 was 0.02 rad2/time3.

5.2.4 Results and Conclusions

Before viewing results for the problem as presented, consider a variation in which just the angle θ is the measured quantity. This was tried first, unsuccessfully. Some runs worked alright, but eventually a run occurred where $\hat{r}$ would shrink to nearly zero, causing Pf to blow up - note all the r's in the denominators of (5-15). At first, scaling was suspected, e.g. a huge force field (Go too large) or an unreasonably small $\hat{r}$(T0), but changing these parameters failed to relieve the problem. Next, measurement accuracy was improved, but the blow-ups continued. Some tinkering with Qfs also was done, but the blow-ups remained. In the end, some performance improvement had resulted but attempts to make the filter work with just θ measurements were called off and labeled unsuccessful. The cause of this failure is the weak coupling between range and angle. Evidently one can know θ very well and know little about r,

even though the system is theoretically observable with only 8 measurements present.

We now return to the problem as presented. Appendix B begins with the printed output for the circular orbit case. This output contains a listing of the user-written routines, printer plots and periodic data for the first run, and statistics for the 50-run study. In addition, Figures B-1 through B-8, which were made by the postprocessor SOFEPL from SOFE's TAPE4 output, depict this filter's performance using ensemble averages.

All available data indicate that $\hat{\underline{X}}f$ tracks $\underline{X}s$ within acceptable bounds. Figures B-1 through B-4 show the average of true estimation error $\bar{e}$ surrounded by $\pm$ Se, the standard deviation computed from e data. The $\bar{e}$ curve in these four figures is approximately zero mean as it should be. Note the agreement between the pair $\bar{e}$,Se as displayed in the SOFEPL curves and in the statistical summary near the end of the listing. This summary certainly provides some quick-look numbers, but the cost in coding of ESTIX and the limited visibility that raw numbers can provide work in favor of using SOFEPL to view results whenever possible.

Figures B-5 through B-8 each contain two curves, a solid line for Se and a broken line for $\overline{\text{SQRT(Pf)}}$. In a con-

servative filter design, $\overline{SQRT(Pf)}$ should generally be somewhat greater than Se so that the filter will be 'open' to the variability in the actual system. Figures B-5 through B-8 show that this dictum has not been consistently satisfied, indicating that, at a minimum, tuning is needed. If tuning is neglected, tracking may diverge for longer orbits.

The study described above was repeated using 10:1 elliptical orbit initial conditions for both the filter and truth models. The results (not shown here) were similar to those for the circular orbit with some improvement in tracking between Se and $\overline{SQRT(Pf)}$.

In a third and final study the truth orbit was the 10:1 ellipse ($\underline{X}$so = 1,0,0,1.34...) while the filter state was initialized with these values: 2.0,0.1,0.2,1.0 . This offset in initial conditions is within the 1-sigma bounds prescribed by Pfo for all states except $\hat{r}$, where the error is roughly 3-sigma. These initial offsets produced a transient in $\overline{e}$ for all four states that died out after two time units had elapsed and four measurements were processed. Other measures of performance were essentially unchanged from the previous 10:1 study.

In summary, the four-state extended filter was able to estimate actual satellite trajectories despite fairly large

initial condition errors. The match between Se and $\overline{\text{SQRT(Pf)}}$ was satisfactory for all four states at all times except for the circular orbit at about five time units. A problem may be surfacing for this case, but it was not pursued here. On balance, Pfo, Rf and Qf appear to be fairly well chosen for the range of missions that were tried.

5.3 Standard Short Test

Appendix C shows three pages of printed output for the linear INS problem with a TF of 61 seconds instead of ten hours. This short run serves as a standard short test case for SOFE. Three propagations (0 to 30, 30 to 60, 60 to 61) and two updates (at 30 and 60) occur in the 61 seconds of the run, thereby exercising all of SOFE's code except the printer plotting and the external trajectory capability. This amounts to a thorough check for a very small expenditure of computer time (less than two seconds).

Since LPRUD, LPRH, LPRK and LPRZR are TRUE, a large amount of output is displayed at update time. Since DTPRNT exceeds 61, the only other output occurs at TO and Tf. Note the final check product at the run's conclusion:

-0.132880246897869 E-143

This number provides a quick and handy check of 'duplicate' test cases.

APPENDIX A

Subroutines and Output for INS Problem

```
      SUBROUTINE AMEND(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
C                                                                AMEND     2
C  USER-WRITTEN SUBROUTINE                                       AMEND     3
C  GCAP/MCAP TEST CASE                                           AMEND     4
C  APPLIES TOTAL FEEDBACK CONTROL                                AMEND     5
C                                                                AMEND     6
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)                        AMEND     7
      DO 100 I=1,NF                                              AMEND     8
         XS(I) = XS(I) - XF(I)                                   AMEND     9
         XF(I) = 0.                                              AMEND    10
  100 CONTINUE                                                   AMEND    11
      RETURN                                                     AMEND    12
C                                                                AMEND    13
      ENTRY AMENDO                                               AMEND    14
C                                                                AMEND    15
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN              AMEND    16
      RETURN                                                     AMEND    17
      END                                                        AMEND    18
                                                                 AMEND    19
```

```
      SUBROUTINE ESTIX(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF)
C                                                                ESTIX     2
C  USER-WRITTEN SUBROUTINE                                       ESTIX     3
C  ESTIX IS CALLED AT DTSTIX INTERVALS TO MAKE OUTPUT PER USER NEEDS  ESTIX  4
C                                                                MAR80    60
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR)                ESTIX     6
      RETURN                                                     ESTIX     7
      ENTRY ESTIXO                                               ESTIX     8
C                                                                ESTIX     9
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN              ESTIX    10
      RETURN                                                     ESTIX    11
      END                                                        ESTIX    12
                                                                 ESTIX    13
```

```
      SUBROUTINE FQGEN(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NZF,NZQ,F,QF)          FEB80      93
C                                                                           FQGEN       3
C  USER-WRITTEN SUBROUTINE                                                  FQGEN       4
C  GCAP/MCAP TEST CASE                                                      FQGEN       5
C  FQGEN CALCULATES THE NON ZERO ELEMENTS OF THE MATRICES F                 FQGEN       6
C  AND QF FOR THE PROPAGATION OF THE COVARIANCE MATRIX PF                   FEB80      94
C                                                                           FQGEN       8
      COMMON /FNOIS/SDWF(2)                                                 FQGEN       9
      COMMON /TCFIL/TAUF(2)                                                 FQGEN      10
      COMMON /TRJCOM/RE,G                                                   FQGEN      11
      DIMENSION XF(NF),XS(NS),F(NZF),QF(NZQ),XTRAJ(NXTJ)                    FEB80      95
C                                                                           FQGEN      13
C  CALCULATE THE TIME VARIANT ELEMENTS OF MATRICES F AND QF                 FEB80      96
      RETURN                                                                FQGEN      15
C                                                                           FQGEN      16
      ENTRY FQGENO                                                          FQGEN      17
C                                                                           FQGEN      18
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN                         FQGEN      19
C  CALCULATE THE TIME INVARIANT ELEMENTS OF MATRICES F AND QF               FEB80      97
      F(1)=1./RE                                                            FQGEN      21
      F(2)=-G                                                               FQGEN      22
      F(3)=1.                                                               FQGEN      23
      F(4)=1./RE                                                            FQGEN      24
      F(5)=1.                                                               FQGEN      25
      F(6)=-1./TAUF(1)                                                      FQGEN      26
      F(7)=-1./TAUF(2)                                                      FQGEN      27
      DO 10 II=1,2                                                          FQGEN      28
         QF(II)=2.0*SDWF(II)*SDWF(II)/TAUF(II)                              FEB80      98
   10 CONTINUE                                                              FQGEN      30
      RETURN                                                                FQGEN      31
      END                                                                   FQGEN      32
```

```
      SUBROUTINE HRZ(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF,              JUN80     235
     *               IMEAS,M,H,RF,ZRES)                                 JUN80     236
C                                                                       HRZ         3
C  USER-WRITTEN SUBROUTINE                                              HRZ         4
C  GCAP/MCAP TEST CASE                                                  HRZ         5
C  FOR EACH MEASUREMENT NUMBER IMEAS, HRZ SUPPLIES:                     HRZ         6
C   1.  MEASURMENT MATRIX ROW VECTOR, H, A LINEAR ARRAY REPRESENTING    HRZ         7
C         H(IMEAS,1),.....,H(IMEAS,NF)                                  HRZ         8
C   2.  MEASUREMENT NOISE VARIANCE, RF, A SCALAR SPECIFYING NOISE LEVEL FEB80     100
C   3.  MEASURMENT RESIDUAL, ZRES, A SCALAR THAT SIMULATES THE          HRZ        10
C         DIFFERENCE BETWEEN THE ACTUAL AND THE PREDICTED MEASUREMENTS  HRZ        11
C  TO DISABLE THE IMEAS'TH MEASUREMENT, MAKE RF(IMEAS) < 0 .            FEB80     101
C                                                                       HRZ        13
      COMMON /RF/RFVCTR(2)                                              FEB80     102
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),H(NF),STD(2)          JUN80     237
C                                                                       HRZ        16
      CALL ZROIZE(1,NF,H)                                               HRZ        17
      RF=RFVCTR(IMEAS)                                                  FEB80     103
      IF (IMEAS-1) 10,10,20                                             FEB80     104
C                                                                       FEB80     105
C  POSITION MEASUREMENT                                                 FEB80     106
 10   H(1)=1.                                                           FEB80     107
      ZRES = XS(1)+XS(8)-XF(1)+GAUSS(0.,STD(IMEAS))                     HRZ        21
      RETURN                                                            HRZ        22
C                                                                       FEB80     108
C  VELOCITY MEASUREMENT                                                 FEB80     109
 20   H(2)=1.                                                           FEB80     110
      ZRES = XS(2)+XS(9)-XF(2)+GAUSS(0.,STD(IMEAS))                     HRZ        24
      RETURN                                                            HRZ        25
C                                                                       HRZ        26
      ENTRY HRZO                                                        HRZ        27
C                                                                       HRZ        28
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN                     HRZ        29
      STD(1)=SQRT(RFVCTR(1))                                            FEB80     111
      STD(2)=SQRT(RFVCTR(2))                                            FEB80     112
      RETURN                                                            HRZ        32
      END                                                               HRZ        33
..
```

```
      SUBROUTINE SNOYS(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)                  SNOYS    2
C                                                                      SNOYS    3
C  USER-WRITTEN SUBROUTINE                                             SNOYS    4
C  GCAP/MCAP TEST CASE                                                 SNOYS    5
C  SNOYS ADDS GAUSSIAN RANDOM SAMPLES TO SPECIFIED SYSTEM              SNOYS    6
C  STATES.  THE PURPOSE IS TO SIMULATE THE ACCUMULATED                 SNOYS    7
C  EFFECT OF PROCESS DRIVING NOISE ON THE SYSTEM OVER THE              SNOYS    8
C  NOISE ACCUMULATION INTERVAL DT.                                     SNOYS    9
C  IN THIS TEST CASE ALL OF THE DRIVING NOISE PROCESSES                SNOYS   10
C  WERE FIRST ORDER MARKOV SO THE APPROPRIATE STANDARD                 SNOYS   11
C  DEVIATION FOR THE NOISE SAMPLE IS                                   SNOYS   12
C        SIGMA*SQRT(2.*DT/TAU)                                         SNOYS   13
C                                                                      SNOYS   14
      COMMON /SNOIS/SDWS(5),SDWS0                                      SNOYS   15
      COMMON /TCSYS/TAUS(5)                                            SNOYS   16
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)                              SNOYS   17
C                                                                      SNOYS   18
      DT=T-TOLD                                                        SNOYS   19
      STDEV=SDWS(1)*SQRT(2.*DT/TAUS(1))                                SNOYS   20
      XS(4)=XS(4)+GAUSS(0.0,STDEV)                                     SNOYS   21
      STDEV=SDWS(2)*SQRT(2.*DT/TAUS(2))                                SNOYS   22
      XS(5)=XS(5)+GAUSS(0.0,STDEV)                                     SNOYS   23
      STDEV=SDWS(3)*SQRT(2.*DT/TAUS(3))                                SNOYS   24
      XS(7)=XS(7)+GAUSS(0.0,STDEV)                                     SNOYS   25
      STDEV=SDWS(4)*SQRT(2.*DT/TAUS(4))                                SNOYS   26
      XS(8)=XS(8)+GAUSS(0.0,STDEV)                                     SNOYS   27
      STDEV=SDWS(5)*SQRT(2.*DT/TAUS(5))                                SNOYS   28
      XS(9)=XS(9)+GAUSS(0.0,STDEV)                                     SNOYS   29
C                                                                      SNOYS   30
      ENTRY SNOYS0                                                     SNOYS   31
C                                                                      SNOYS   32
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN                    SNOYS   33
      TOLD=T                                                           SNOYS   34
      RETURN                                                           SNOYS   35
      END                                                              SNOYS   36


      SUBROUTINE TRAJO(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)                  JUN80  238
C                                                                      TRAJ     3
C  USER-WRITTEN SUBROUTINE                                             TRAJ     4
C  GCAP/MCAP TEST CASE                                                 TRAJ     5
C  IN THIS TEST CASE, TRAJO SERVES ONLY TO SET SOME CONSTANTS          JUN80  239
C                                                                      TRAJ     7
      COMMON /TRJCOM/RE,G                                              TRAJ     8
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ)                              TRAJ     9
C                                                                      JUN80  240
      RE=2.09E+7                                                       TRAJ    15
      G=32.2                                                           TRAJ    16
      RETURN                                                           TRAJ    17
      END                                                              TRAJ    18
```

```
      SUBROUTINE USRIN                                                  USRIN      2
C                                                                       USRIN      3
C   USER-WRITTEN SUBROUTINE                                             USRIN      4
C   GCAP/MCAP TEST CASE                                                 USRIN      5
C   THIS ROUTINE READS AND PRINTS THE USER SUPPLIED CONSTANTS.          USRIN      6
C                                                                       USRIN      7
      COMMON /FNOIS/SDWF(2)                                             USRIN      8
      COMMON /RF/RFVCTR(2)                                              FEB80    113
      COMMON /TCFIL/TAUF(2)                                             USRIN     10
      COMMON /SNOIS/SDWS(5),SDWS0                                       USRIN     11
      COMMON /TCSYS/TAUS(5)                                             USRIN     12
C                                                                       USRIN     13
      NAMELIST/INF/TAUF,SDWF,RFVCTR                                     FEB80    114
      NAMELIST/INS/TAUS,SDWS,SDWS0                                      USRIN     15
C                                                                       USRIN     16
      READ (5,INF)                                                      USRIN     17
      WRITE (6,100) TAUF,SDWF,RFVCTR                                    FEB80    115
      READ (5,INS)                                                      USRIN     19
      WRITE (6,200) TAUS,SDWS,SDWS0                                     OCT79    137
      RETURN                                                            USRIN     21
C                                                                       FEB80    116
 100  FORMAT('1'5(/)T10'FILTER MODEL DATA-BASE FROM NAMELIST INF:'      OCT79    138
     * //T15'TIME CONSTANTS, TAUF '2G14.7                               OCT79    139
     * //T15'STD DEV'S SDWF FOR FORMING WHITE NOISE STRENGTHS IN FQGEN ' OCT79    140
     * 2G14.7//T15'MEASUREMENT NOISE VARIANCES, RF '2G14.7 )            OCT79    141
 200  FORMAT(9(/)T10'TRUTH  MODEL DATA-BASE FROM NAMELIST INS:'         OCT79    142
     * //T15'TIME CONSTANTS, TAUS '5G14.7                               OCT79    143
     * //T15'STD DEV'S SDWS AND SDWS0 FOR FORMING PROCESS NOISE SAMPLES' OCT79    144
     * /T20,6G14.7 )                                                    OCT79    145
      END                                                               USRIN     22
```

```
      SUBROUTINE XFDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF,XDOT)           JUN80     241
C                                                                           XFDOT       3
C  USER-WRITTEN SUBROUTINE                                                  XFDOT       4
C  GCAP/MCAP TEST CASE                                                      XFDOT       5
C  XFDOT COMPUTES THE FILTER DERIVATIVES                                    XFDOT       6
C                                                                           XFDOT       7
      COMMON /TCFIL/TAUF(2)                                                 XFDOT       8
      COMMON /TRJCOM/RE,G                                                   XFDOT       9
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),XDOT(NF)                  JUN80     242
C                                                                           JUN80     243
C    XDOT CALCULATIONS                                                      XFDOT      11
      XDOT(1) = XF(2)                                                       XFDOT      12
      XDOT(2) = -XF(3)*G + XF(5)                                            XFDOT      13
      XDOT(3) =  XF(2)/RE + XF(4)                                           XFDOT      14
      XDOT(4) = -XF(4)/TAUF(1)                                              XFDOT      15
      XDOT(5) = -XF(5)/TAUF(2)                                              XFDOT      16
      RETURN                                                                XFDOT      17
C                                                                           XFDOT      18
      ENTRY XFDOTO                                                          XFDOT      19
C                                                                           XFDOT      20
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN                         XFDOT      21
      RETURN                                                                XFDOT      22
      END                                                                   XFDOT      23


      SUBROUTINE XSDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,XDOT)                  XSDOT       2
C                                                                           XSDOT       3
C  USER-WRITTEN SUBROUTINE                                                  XSDOT       4
C  GCAP/MCAP TEST CASE                                                      XSDOT       5
C  XSDOT COMPUTES DERIVATIVES FOR THE TRUTH MODEL                           JUN80     244
C        XSDOT = F(XS,T)                                                    XSDOT       7
C                                                                           XSDOT       8
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),XDOT(NS)                          XSDOT       9
      COMMON /TCSYS/TAUS(5)                                                 XSDOT      10
      COMMON /SNOIS/SDWS(5),SDWS0                                           XSDOT      11
      COMMON /TRJCOM/RE,G                                                   XSDOT      12
      XDOT(1) = XS(2)                                                       XSDOT      13
      XDOT(2) = -G*XS(3) + XS(5) + XS(7)                                    XSDOT      14
      XDOT(3) = XS(2)/RE + XS(4) + XS(6)                                    XSDOT      15
      XDOT(4) = -XS(4)/TAUS(1)                                              XSDOT      16
      XDOT(5) = -XS(5)/TAUS(2)                                              XSDOT      17
      XDOT(6) = 0.                                                          XSDOT      18
      XDOT(7) = -XS(7)/TAUS(3)                                              XSDOT      19
      XDOT(8) = -XS(8)/TAUS(4)                                              XSDOT      20
      XDOT(9) = -XS(9)/TAUS(5)                                              XSDOT      21
      RETURN                                                                XSDOT      22
C                                                                           XSDOT      23
      ENTRY XSDOTO                                                          XSDOT      24
C                                                                           XSDOT      25
C  INITIALIZATION:  OCCURS AT START OF EACH NEW RUN                         XSDOT      26
      XS(6) = GAUSS(0.,SDWS0)                                               XSDOT      27
      RETURN                                                                XSDOT      28
      END                                                                   XSDOT      29
```

SHM22GK

CORRECTION IDENTS ARE LISTED IN CHRONOLOGICAL ORDER OF INSERTION

YANKSSS	SOFE	ADVANS	ASYSP	DERIV	FPPPFT	GAUSS	GETPF
GETX	GOPLOT	ICSEVU	ICSICU	INTERP	KUTMER	MOVE	NZRCIO
OUT	PAGCON	PLCC	PLPR	PRDG	PRLT	PRX	PSQRT
SOFEBD	SOFEIN	SPAN	SPLITA	SQRS	VALDTA	ZROIZE	AMEND
ESTIX	FOGEN	HRZ	SNOYS	TRAJ	USRIN	XFDOT	XSDOT
JAN79	MAR79	OCT79	JAN80	FEB80	MAR80	XSPLUS	JUN80

DECKS ARE LISTED IN THE ORDER OF THEIR OCCURRENCE ON A NEW PROGRAM LIBRARY IF ONE IS CREATED BY THIS UPDATE

YANKSSS	SOFE	ADVANS	ASYSP	DERIV	FPPPFT	GAUSS	GETPF
GETX	GOPLOT	ICSEVU	ICSICU	INTERP	KUTMER	MOVE	NZRCIO
OUT	PAGCON	PLCC	PLPR	PRDG	PRLT	PRX	PSQRT
SOFEBD	SOFEIN	SPAN	SPLITA	SQRS	VALDTA	XSPLUS	ZROIZE
AMEND	ESTIX	FOGEN	HRZ	SNOYS	TRAJ	USRIN	XFDOT
XSDOT							

DECK LIST AS WRITTEN TO SEQUENTIAL NEWPL

YANKSSS	SOFE	ADVANS	ASYSP	DERIV	FPPPFT	GAUSS	GETPF
GETX	GOPLOT	ICSEVU	ICSICU	INTERP	KUTMER	MOVE	NZRCIO
OUT	PAGCON	PLCC	PLPR	PRDG	PRLT	PRX	PSQRT
SOFEBD	SOFEIN	SPAN	SPLITA	SQRS	VALDTA	XSPLUS	ZROIZE
AMEND	ESTIX	FOGEN	HRZ	SNOYS	TRAJ	USRIN	XFDOT
XSDOT							

```
SOFE     ADVANS   ASYSP    DERIV    FPPPFT   GAUSS    GETPF    GETX
GOPLOT   ICSEVU   ICSICU   INTERP   KUTMER   MOVE     NZRCID   OUT
PAGCON   PLCC     PLPR     PROG     PRLT     PRX      PSORT    SOFEBD
SOFFIN   SPAN     SPLITA   SORS     VALDTA   XSPLUS   ZROIZE   AMEND
ESTIX    FQGEN    HRZ      SNOYS    TRAJ     USRIN    XFDOT    XSOUT
```

THIS UPDATE REQUIRED 346008 WORDS OF CORE.

RUN DATE AND TIME : 07/17/80 16.07.13.

```
************************************************ S O F E   B E G I N S ************************************************

                                   S O F E

                  A GENERALIZED MONTE CARLO SIMULATION FOR
           THE DESIGN AND PERFORMANCE ANALYSIS OF MULTISENSOR SYSTEMS
            WHERE DATA IS COMBINED USING KALMAN FILTER TECHNIQUES



PROBLEM TITLE  :   GCAP/MCAP SINGLE AXIS INS --- A STANDARD LONG TEST FOR "SOFE"



RUN DATE AND TIME  :   07/17/80  16.07.13.
```

```
             COMPOSITE OF INPUT DATA AND DEFAULT VALUES IN THE 'PRDATA' NAMELIST GROUP
-----------------------------------------------------------------------------------------------
DIMENSIONS AND SIZES:
     NF =   5        NS =   9        N =   2       NZF =    7       NZO =    2       NXTJ =    0

EXTERNAL TRAJECTORY:
     LXTJ =  F

BEGIN AND END TIME OF EACH RUN:
     TO =  0.                   TF =   36000.

TIME WHEN UPDATES MAY BEGIN:
     TMEASO =  0.

TIME INTERVAL BETWEEN UPDATES, PRINTS, STORAGE FOR CALCOMP AND PRINTER PLOTS, CALLS TO ESTIX AND SNOYS:
     DTMEAS =   30.000             DTPRNT =    3600.0             DTCCPL =    .10000E+10
     DTPRPL =   360.00             DTSTIX =    .10000E+10         DTNOYS =    30.000

PARAMETERS GOVERNING PRINTED OUTPUT:
     LPR =     T        LPRXS =  T        LPRXF =  T        LPRUG =  T        LPRLT =  F        LPRJO =  F
     LPRZR =  F         LPRH =   F        LPRK =   F        LPRXTJ = F        IPGSIZ =  55      IPRRUN =      1

PARAMETER GOVERNING DATA STORAGE FOR POSTRUN CALCOMP PLOTTING:
     LCC =  F

PARAMETERS GOVERNING PRINTER PLOTTING:
     LPP =  T        LPPLD =  F        LPPUP =  F

INTEGRATION CONTROL PARAMETERS:
     MODE =  1       TOLER =    .10000E-03       HMAX =    .10000E+10       HMIN =    .10000E-03       HO =    .10000E-01

CONTINUATION FLAG, RANDOM NUMBER SEED, NUMBER OF RUNS:
     ICONT =  0      ISEED =    77       IPASS =      1
-----------------------------------------------------------------------------------------------


THE FOLLOWING    7 ROW-COLUMN PAIRS LOCATE THE NONZERO ELEMENTS OF THE SPARSE MATRIX F
  1.      1   2          2.      2   3          3.      2   5          4.      3   2          5.      3   4          6.      4   4
  7.      5   5


THE FOLLOWING    2 ROW-COLUMN PAIRS LOCATE THE NONZERO ELEMENTS OF THE SPARSE MATRIX QF
  1.      4   4          2.      5   5
```

STRUCTURE OF UNLABELED COMMON AREA 'A'

VECTOR OR MATRIX	LENGTH	FIRST WORD LOCATION
GENERAL WORKING SPACE	NF+2*M	1
COMPOSITE OF (XS,XF,PF), Y	NALL	10
SYSTEM STATE VECTOR, XS	NS	10
FILTER STATE VECTOR, XF	NF	19
FILTER COVARIANCE, PF	NTR	24
SQUARE ROOT OF PF, S	NTR	39
INDICES OF NONZEROES IN F	2*NZF	54
INDICES OF NONZEROES IN QF	2*NZQ	68
VALUES OF NONZEROES IN F	NZF	72
VALUES OF NONZEROES IN QF	NZQ	79
INITIAL CONDITIONS FOR XS	NS	81
INITIAL CONDITIONS FOR XF	NF	90
INITIAL CONDITIONS FOR PF	NTR	95
DERIVATIVE OF Y	NALL	110
DERIVATIVE OF XS	NS	110
DERIVATIVE OF XF	NF	119
DERIVATIVE OF PF	NTR	124
D.E. WORKING SPACE	4*NALL	139
MEASUREMENT SENSITIVITY, H	NF	255
PRODUCT OF S AND H	NF	260
KALMAN GAIN	NF	265
XTRAJ TIMES	3	270
XTRAJ DATA	3*NXTJ	273
XTRAJ INTERPOLATED DATA	4*NXTJ	273
XTRAJ INTERPOLATION COEFFICIENTS	6*NXTJ	273

NOTE:

NS	= DIMENSION OF SYSTEM STATE	=	9
NF	= DIMENSION OF FILTER STATE	=	5
M	= NUMBER OF MEASUREMENTS PER UPDATE	=	2
NTR	= SIZE OF TRIANGULAR PART OF PF	=	15
NALL	= DIMENSION OF COMPOSITE STATE Y (NS+NF+NTR)	=	29
NZF	= NUMBER OF NONZEROES IN F	=	7
NZQ	= NUMBER OF NONZEROES IN QF	=	2
NXTJ	= NUMBER OF VARIABLES ON XTRAJ TAPE	=	0

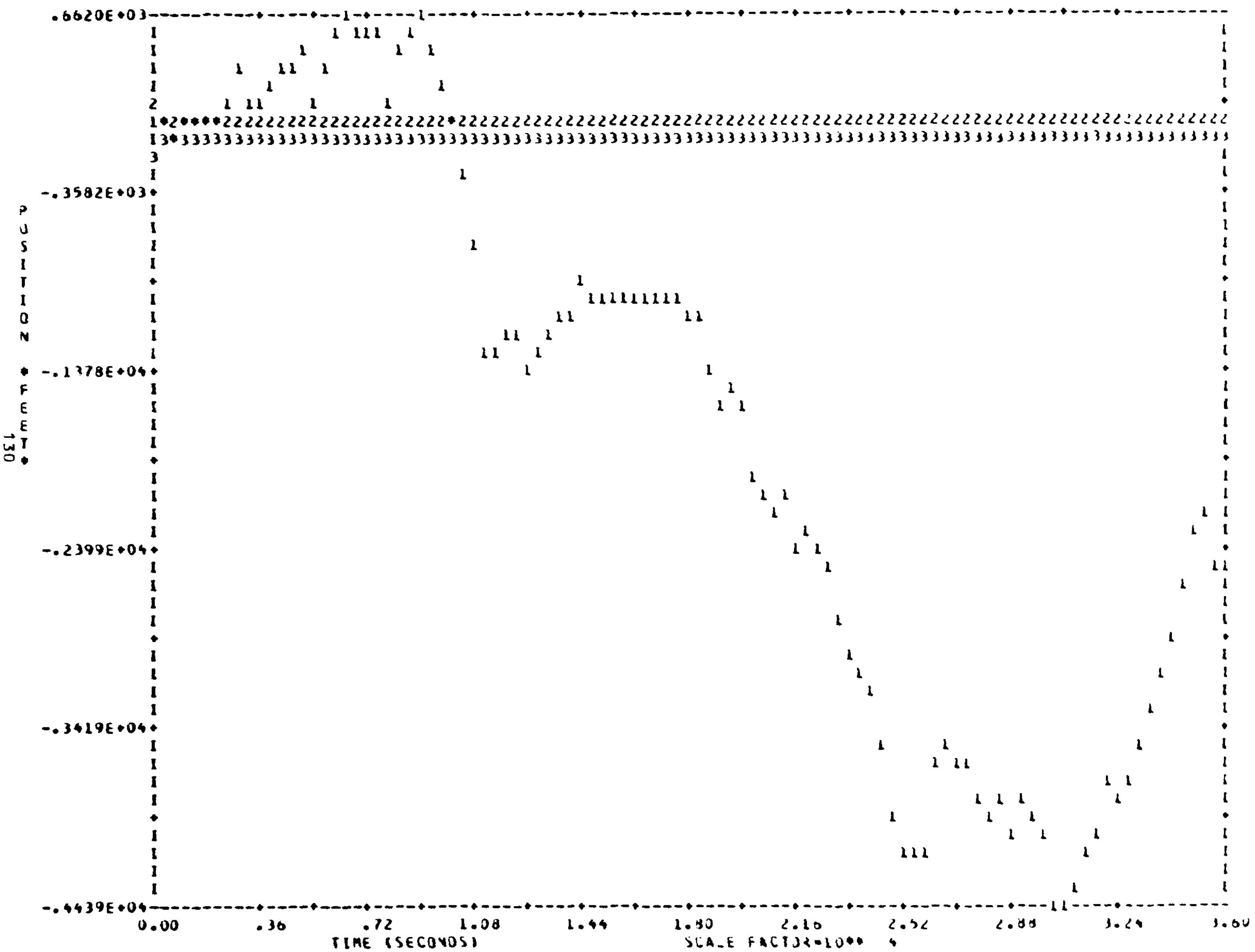
AVAILABLE SPACE IN BLANK COMMON 'A' 1000
SPACE IN 'A' REQUIRED BY USER'S PROBLEM 272

```
FILTER MODEL DATA-BASE FROM NAMELIST INF:

     TIME CONSTANTS, TAUF   3600.000      300.0000

     STD DEV'S SDWF FOR FORMING WHITE NOISE STRENGTHS IN FOGEN   .4848000E-07  .6442000E-02

     MEASUREMENT NOISE VARIANCES, RF   10000.00      .2500000


TRUTH  MODEL DATA-BASE FROM NAMELIST INS:

     TIME CONSTANTS, TAUS   3600.000      300.0000      3600.000      1800.000      300.0000

     STD DEV'S SDWS AND SDWSO FOR FORMING PROCESS NOISE SAMPLES
           .4848000E-07  .6442000E-02  .3220000E-02  300.0000      .5000000      .2420000E-07
```

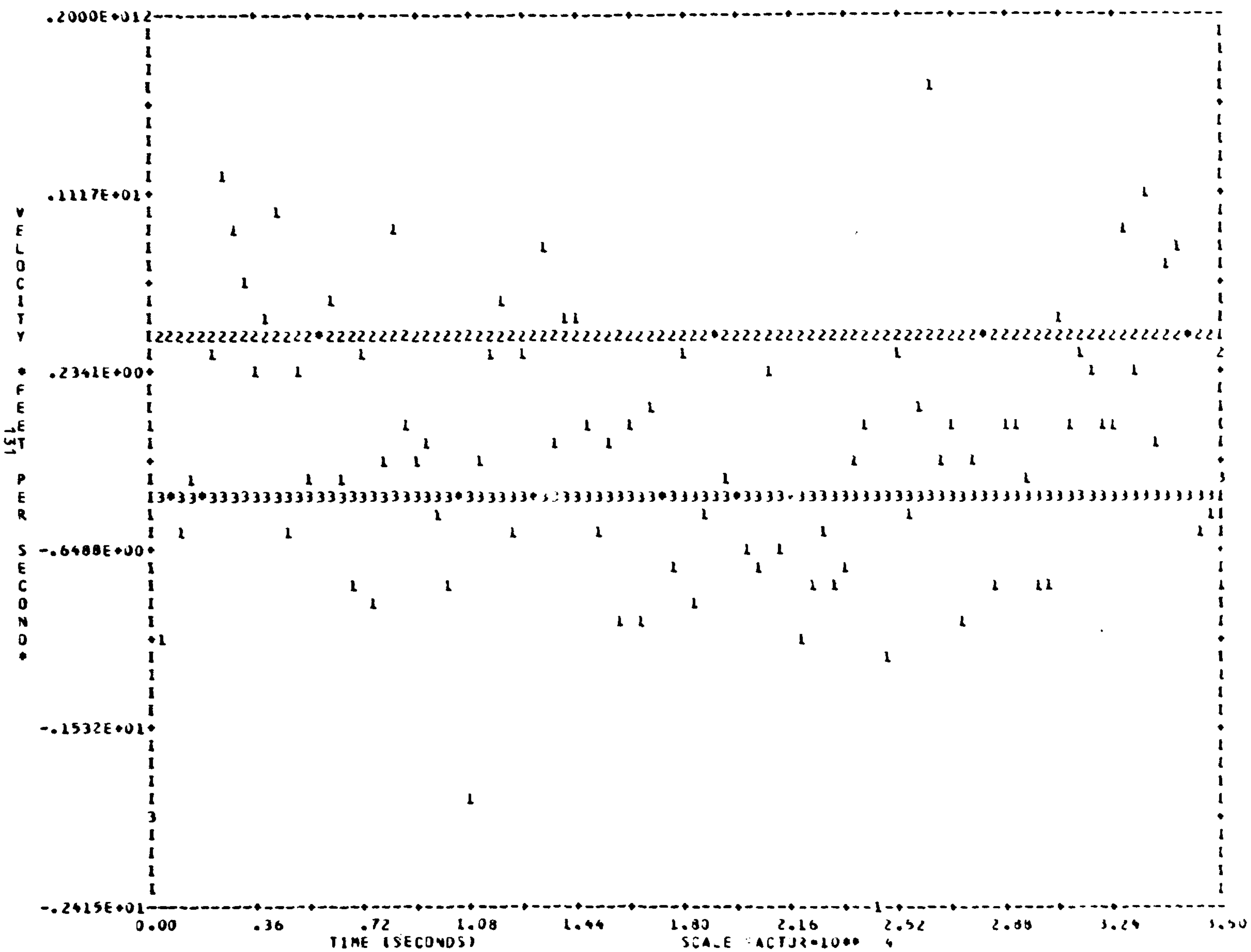
```
STATE VECTOR XSO   AT T =  0.
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
    6. -8.289704E-09      7.  0.                8.  0.                9.  0.
STATE VECTOR XFO   AT T =  0.
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XFO   AT T =  0.
    1.   120.000          2.   2.00000          3.  1.745279E-03      4.  4.847680E-08      5.  6.439720E-03

STATE VECTOR XS    AT T =   3600.0
    1.   192.546          2.   .199353          3.  1.355231E-04      4.  3.648895E-08      5.  3.285838E-03
    6. -8.289704E-09      7.  4.357614E-03      8.  -206.849          9.   .272541
STATE VECTOR XF    AT T =   3600.0
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XF    AT T =   3600.0
    1.   9.14035          2.   .406872          3.  1.018782E-04      4.  4.482086E-08      5.  5.217617E-03

STATE VECTOR XS    AT T =   7200.0
    1.   603.926          2.   .304109          3.  1.186821E-04      4. -1.529383E-08      5.  3.313791E-03
    6. -8.289704E-09      7.  6.832210E-04      8.  -227.982          9.  2.555253E-02
STATE VECTOR XF    AT T =   7200.0
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XF    AT T =   7200.0
    1.   6.45909          2.   .406795          3.  1.012945E-04      4.  4.278351E-08      5.  5.212017E-03

STATE VECTOR XS    AT T =   10800.
    1.  -648.019          2.  -1.87212          3. -8.656970E-05      4. -3.333607E-09      5. -4.111416E-03
    6. -8.289704E-09      7.  3.073407E-03      8.   149.885          9.   1.51449
STATE VECTOR XF    AT T =   10800.
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XF    AT T =   10800.
    1.   5.27270          2.   .406735          3.  1.008719E-04      4.  4.270777E-08      5.  5.208352E-03

STATE VECTOR XS    AT T =   14400.
    1.  -900.302          2.   .501770          3.  1.322036E-04      4.  5.324628E-08      5.  2.282655E-03
    6. -8.289704E-09      7.  1.058361E-03      8.  -464.949          9.   .367245
STATE VECTOR XF    AT T =   14400.
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XF    AT T =   14400.
    1.   4.56581          2.   .406732          3.  1.008466E-04      4.  4.270555E-08      5.  5.208112E-03

STATE VECTOR XS    AT T =   18000.
    1.  -1060.46          2.   .311723          3. -1.233892E-05      4. -5.128883E-10      5.  9.762179E-03
    6. -8.289704E-09      7. -8.868768E-04      8.  -150.471          9.  -.530913
STATE VECTOR XF    AT T =   18000.
    1.  0.                2.  0.                3.  0.                4.  0.                5.  0.
SIGMAS  FOR  XF    AT T =   18000.
    1.   4.08352          2.   .406732          3.  1.008463E-04      4.  4.270483E-08      5.  5.208110E-03
```

```
STATE VECTOR XS    AT T =   21600.
    1.  -2362.58             2.  -.370000             3. -1.084920E-04          4. -6.494564E-08          5.  1.044714E-03
    6. -8.289704E-09         7. -3.565460E-03         8.  -11.0535              9.    .639298
STATE VECTOR XF    AT T =   21600.
    1.  0.                   2.  0.                   3.  0.                    4.  0.                    5.  0.
SIGMAS  FOR  XF    AT T =   21600.
    1.   3.72757             2.   .406732             3.  1.008452E-04          4.  4.270481E-08          5.  5.208108E-03

STATE VECTOR XS    AT T =   25200.
    1.  -4121.04             2.   .321117             3. -3.462368E-04          4. -1.760325E-08          5.  2.100151E-03
    6. -8.289704E-09         7. -1.917727E-03         8.  -31.8453              9.    .426943
STATE VECTOR XF    AT T =   25200.
    1.  0.                   2.  0.                   3.  0.                    4.  0.                    5.  0.
SIGMAS  FOR  XF    AT T =   25200.
    1.   3.45096             2.   .406732             3.  1.008461E-04          4.  4.270481E-08          5.  5.208108E-03

STATE VECTOR XS    AT T =   28800.
    1.  -4027.77             2.  1.067868E-02         3. -2.219014E-04          4. -6.242135E-08          5. -1.581844E-03
    6. -8.289704E-09         7. -7.410110E-04         8.  -485.118              9.   -.384158
STATE VECTOR XF    AT T =   28800.
    1.  0.                   2.  0.                   3.  0.                    4.  0.                    5.  0.
SIGMAS  FOR  XF    AT T =   28800.
    1.   3.22800             2.   .406732             3.  1.008451E-04          4.  4.270481E-08          5.  5.208109E-03

STATE VECTOR XS    AT T =   32400.
    1.  -3804.59             2.  9.423234E-03         3. -2.257742E-04          4. -4.031490E-08          5.  3.911350E-03
    6. -8.289704E-09         7. -2.468540E-03         8.  -23.0084              9.    .395512
STATE VECTOR XF    AT T =   32400.
    1.  0.                   2.  0.                   3.  0.                    4.  0.                    5.  0.
SIGMAS  FOR  XF    AT T =   32400.
    1.   3.04333             2.   .406732             3.  1.008451E-04          4.  4.270451E-08          5.  5.208108E-03

STATE VECTOR XS    AT T =   36000.
    1.  -2545.14             2.  -.815191             3. -1.761397E-04          4. -4.453011E-08          5. -6.022555E-03
    6. -8.289704E-09         7. -3.163159E-03         8.  -13.3434              9.    .929980
STATE VECTOR XF    AT T =   36000.
    1.  0.                   2.  0.                   3.  0.                    4.  0.                    5.  0.
SIGMAS  FOR  XF    AT T =   36000.
    1.   2.88592             2.   .315522             3.  1.002334E-04          4.  4.260136E-08          5.  4.593030E-03

RUN NUMBER      1 COMPLETE AT T =    36000.00

KALMAN FILTER SIMULATION COMPLETE AFTER RUN      1
```

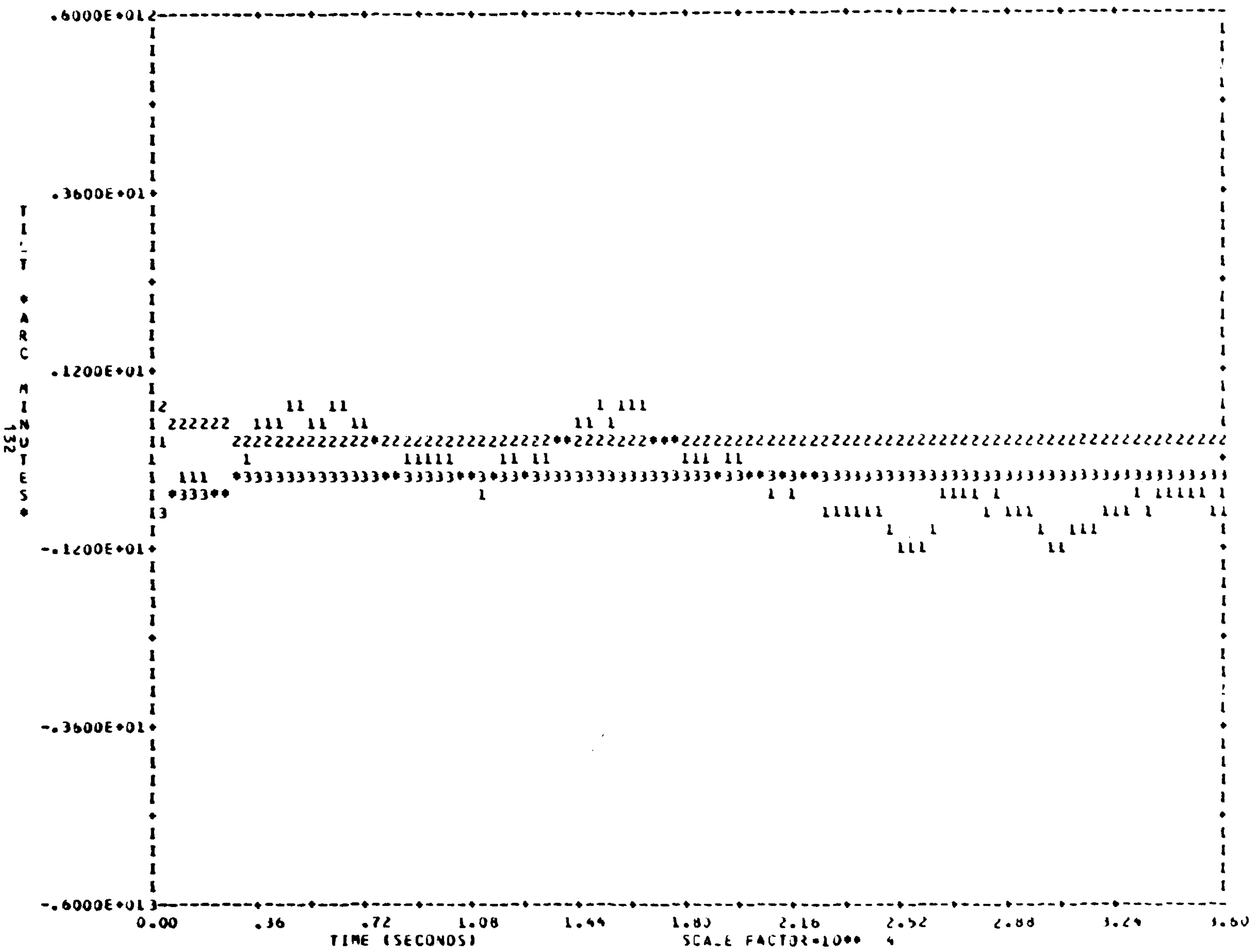
PLOT PARAMETER SET 1,1,1



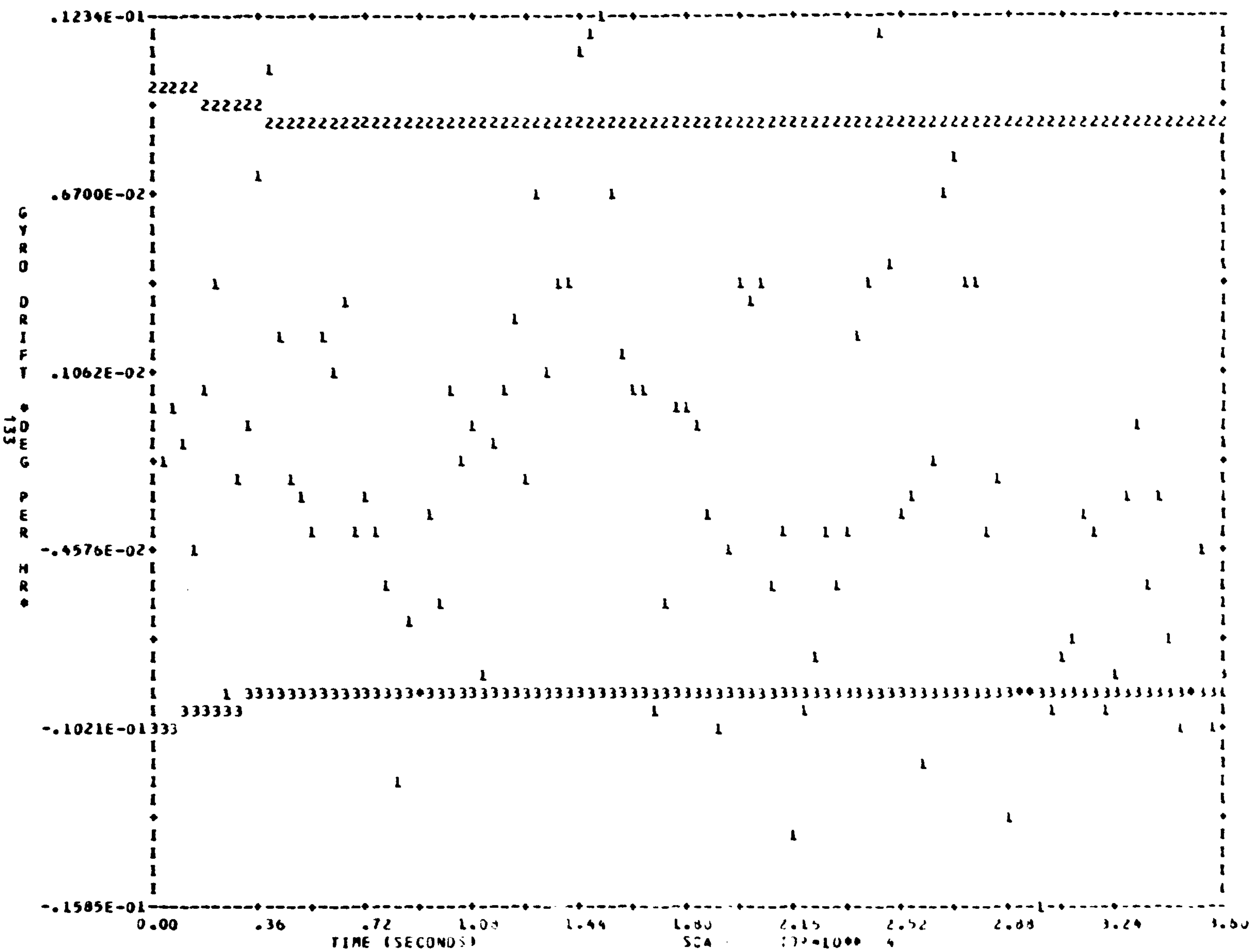
PLOT PARAMETER SET 2,2,2



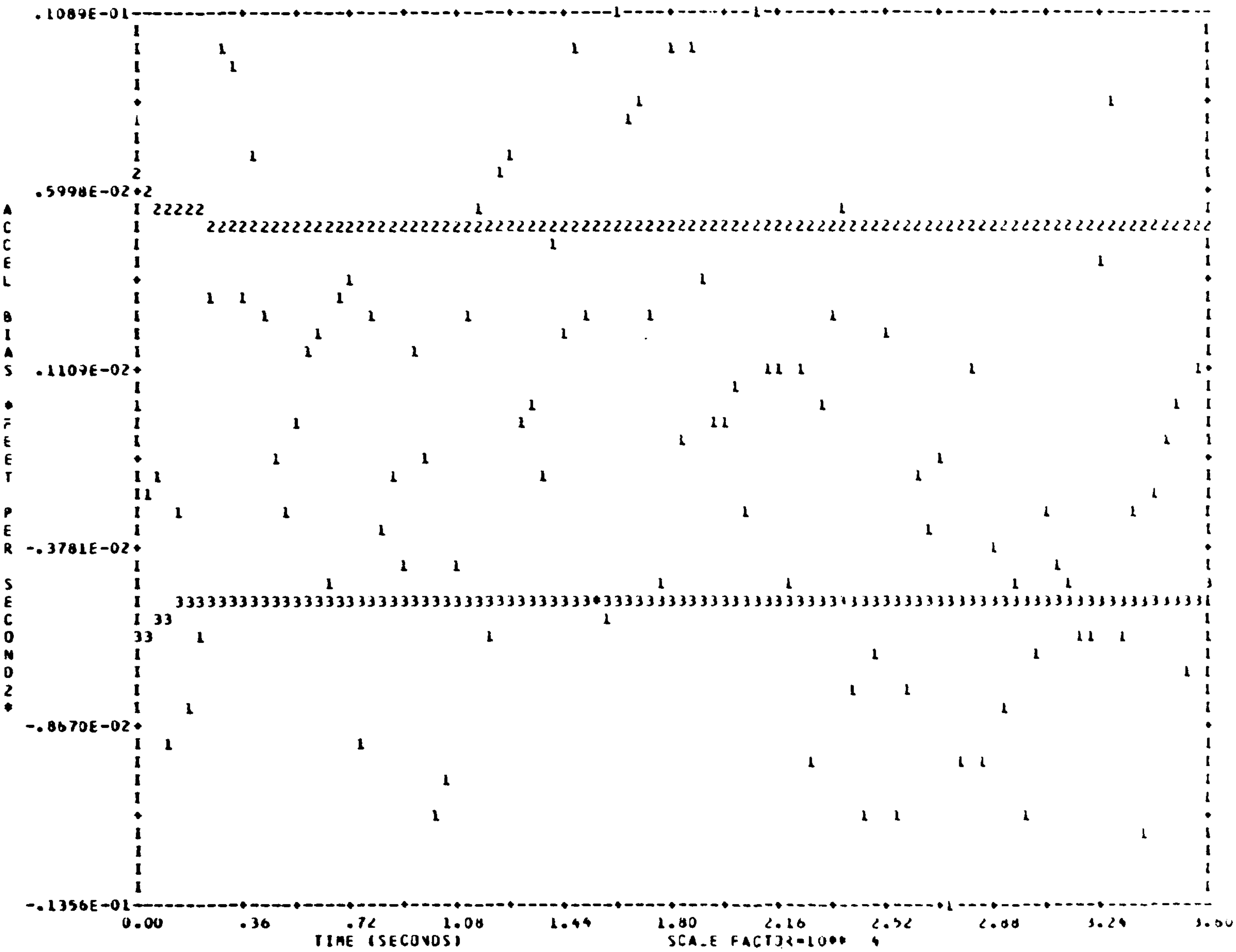
PLOT PARAMETER SET 3,3,3



PLOT PARAMETER SET 4,4,4



PLOT PARAMETER SET 5,5,5



```
FINAL CHECK PRODUCT IS   .548356384797353-134

************************************************ S O F E   F I N I S H E D **************************************************
```

```
  CSA     NOS/BE L499D       L499D-CMR1 06/16/80
16.01.21.SHMGKTJ  FROM     /GK
16.01.22.IP  00000192 WORDS - FILE INPUT  , DC 04
15.01.22.SHM,T35,CM75000.V720130,MUSICK
16.01.27.*
16.01.27.* STANDARD LONG TEST FOR 'SOFE'.
16.01.27.*
16.01.27.* ATTACH AND COMPILE BASIC SOFE WITH
16.01.27.* USER-WRITTEN ROUTINES APPENDED.
16.01.28.ATTACH,OLDPL,SOFE,CY=999,ID=SHM,SN=AFAL,
16.01.28.MR=1.
16.01.28.UPDATE,F,C=COMPILE,O=OUTPUT.
16.01.47. UPDATE COMPLETE.
16.01.47.FTN,I=COMPILE,L=0.
16.06.58.      4.350 CP SECONDS COMPILATION TIME
16.06.58.RETURN,OLDPL,COMPILE.
15.06.59.*
16.06.59.* ATTACH CARD INPUT DATA AND RUN SOFE.
15.06.59.ATTACH,TAPE5,SOFEDATA,CY=222,ID=SHM,SN=A
16.06.59.FA ,MR=1.
16.07.00.LGO.
16.07.36.    STOP  SOFE DONE
15.07.36.     6.398 CP SECONDS EXECUTION TIME
16.07.37.OP  00004992 WORDS - FILE OUTPUT , DC 40
16.07.37.MS     7158  WORDS  (   46592 MAX USED)
16.07.37.CPA      11.758 SEC.         9.582 ADJ.
16.07.37.IO       68.885 SEC.        20.390 ADJ.
16.07.37.CM     1638.986 KWS.         7.720 ADJ.
16.07.37.CRUS                        37.693
16.07.37.CUST                  $      2.48
16.07.37.PP      203.254 SEC.     DATE 07/17/80
16.07.37.EJ  END OF JOB, GK  V720130.
```

APPENDIX B

Subroutines and Output for Orbit Problem

SHMØRGK

```
/////     *COMPILE SOFE.ZROIZE


          MODIFICATIONS / CONTROL CARDS



               CORRECTION IDENTS ARE LISTED IN CHRONOLOGICAL ORDER OF INSERTION

          YANKSSS   SOFE      ADVANS    ASYSP     DERIV     FPPPFT    GAUSS     GETPF
          GETX      GOPLOT    ICSEVU    ICSICU    INTERP    KUTMER    MOVE      NZRCIO
          OUT       PAGCON    PLCC      PLPR      PRDG      PRLT      PRX       PSQRT
          SOFEBD    SOFEIN    SPAN      SPLITA    SQRS      VALDTA    ZROIZE    AMEND
          ESTIX     FQGEN     HRZ       SNOYS     TRAJ      USRIN     XFDOT     XSDOT
          JAN79     MAR79     OCT79     JAN80     FEB80     MAR80     XSPLUS    JUN80


           DECK LIST AS READ FROM OLDPL PLUS ADDED NEW DECKS

          YANKSSS   SOFE      ADVANS    ASYSP     DERIV     FPPPFT    GAUSS     GETPF
          GETX      GOPLOT    ICSEVU    ICSICU    INTERP    KUTMER    MOVE      NZRCIO
          OUT       PAGCON    PLCC      PLPR      PRDG      PRLT      PRX       PSQRT
          SOFEBD    SOFEIN    SPAN      SPLITA    SQRS      VALDTA    XSPLUS    ZROIZE
          AMEND     ESTIX     FQGEN     HRZ       SNOYS     TRAJ      USRIN     XFDOT
          XSDOT


DECKS ARE LISTED IN THE ORDER OF THEIR OCCURRENCE ON A NEW PROGRAM LIBRARY IF ONE IS CREATED BY THIS UPDATE

          YANKSSS   SOFE      ADVANS    ASYSP     DERIV     FPPPFT    GAUSS     GETPF
          GETX      GOPLOT    ICSEVU    ICSICU    INTERP    KUTMER    MOVE      NZRCIO
          OUT       PAGCON    PLCC      PLPR      PRDG      PRLT      PRX       PSQRT
          SOFEBD    SOFEIN    SPAN      SPLITA    SQRS      VALDTA    XSPLUS    ZROIZE
```

```
   DECK LIST AS WRITTEN TO SEQUENTIAL NEWPL

        YANK$$$   SOFE      ADVANS    ASYSP     DERIV     FPPPFT    GAUSS     GETPF
        GETX      GOPLOT    ICSEVU    ICSICU    INTERP    KUTMER    MOVE      NZRCIO
        OUT       PAGCON    PLCC      PLPR      PRDG      PRLT      PRX       PSQRT
        SOFEBD    SOFEIN    SPAN      SPLITA    SQRS      VALDTA    XSPLUS    ZROIZE
        AMEND     ESTIX     FQGEN     HRZ       SNOYS     TRAJ      USRIN     XFDOT
        XSDOT


        DECKS WRITTEN TO COMPILE FILE

        SOFE      ADVANS    ASYSP     DERIV     FPPPFT    GAUSS     GETPF     GETX
        GOPLOT    ICSEVU    ICSICU    INTERP    KUTMER    MOVE      NZRCIO    OUT
        PAGCON    PLCC      PLPR      PRDG      PRLT      PRX       PSQRT     SOFEBD
        SOFEIN    SPAN      SPLITA    SQRS      VALDTA    XSPLUS    ZROIZE

 THIS UPDATE REQUIRED  346008 WORDS OF CORE.
```

```
      SUBROUTINE AMEND(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
C
C     USER-WRITTEN SUBROUTINE
C     NONLINEAR SATELLITE ORBIT PROBLEM
C     NO FEEDBACK CONTROL REQUIRED
C
      RETURN
      ENTRY AMENDO
      RETURN
      END
```

```
      SUBROUTINE ESTIX(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF)
C
C  USER-WRITTEN SUBROUTINE.
C  NONLINEAR SATELLITE ORBIT PROBLEM.
C  ESTIX STORES E = XS-XF DIFFERENCES FOR EACH OF FOUR STATES
C  ACROSS SEVERAL RUNS AND COMPUTES THE MEAN AND VARIANCE OF E FOR
C  THESE STATES AT THE END OF THE PROBLEM. ESTIX ALSO STORES SIGMA DATA
C  FROM PF AND COMPUTES SIGMA MEANS AT THE END OF THE PROBLEM.
C  THE ARRAYS WERE DIMENSIONED TO HANDLE 11 SAMPLE POINTS
C  IN EACH PASS SO DTSTIX WAS SET TO TF/10.
C
      COMMON /ICOM/ICONT,ISEED,IPASS
      COMMON /LCOM/LPR,LPRXS,LPRXF,LPRDG,LPRLT,LPRUD,LPRZR,LPRH,
     *             LPRK,LPRXTJ,LCC,LPP,LPPLD,LPPUP,LPAGE
      DIMENSION XF(NF),XS(NS),PF(NTR),SUMX(4,11),SUMX2(4,11)
      DIMENSION XAVG(11),STDEV(11),SUMPFD(4,11),PFDIAG(11),TSAMPL(11)
      LOGICAL LPAGE
      DATA NSAMPLS/0/
C
C  SAVE SUM OF THE DIFFERENCES , SUM OF THE DIFFERENCES
C  SQUARED AND SIGMA SUMS AT TO AND AT THE NEXT 10 DTSTIX INTERVALS.
C
   10 IF (IPASS.EQ.1) RETURN
      IF (IPNT.GE.11) RETURN
      IPNT=IPNT+1
      TSAMPL(IPNT)=T
      J=0
      DO 30 ISTATE=1,4
         DIFX=XS(ISTATE)-XF(ISTATE)
         SUMX(ISTATE,IPNT)=SUMX(ISTATE,IPNT)+DIFX
         SUMX2(ISTATE,IPNT)=SUMX2(ISTATE,IPNT)+DIFX*DIFX
         J=J+ISTATE
         SUMPFD(ISTATE,IPNT)=SUMPFD(ISTATE,IPNT)+SQRT(PF(J))
   30 CONTINUE
C
C  IF AT PROBLEM END, COMPUTE AND PRINT TRUE ERROR MEANS AND STANDARD
C  DEVIATIONS AS WELL AS AVERAGES FOR SIGMAS.
C
      IF (IPNT.LT.11 .OR. IRUN.NE.IPASS) RETURN
      IF (LPAGE) CALL PAGCON(32,0)
      WRITE(6,1000) NSAMPLS
      WRITE(6,1010) TSAMPL
      DO 50 ISTATE=1,4
         DO 40 IPT=1,11
            XAVG(IPT)=SUMX(ISTATE,IPT)/NSAMPLS
   40    CONTINUE
      WRITE(6,1020) ISTATE,(XAVG(IPT),IPT=1,11)
   50 CONTINUE
C
      IF (NSAMPLS.LT.2) RETURN
      WRITE(6,1030)
      DO 70 ISTATE=1,4
         DO 60 IPT=1,11
            VAR=(SUMX2(ISTATE,IPT)
     *           -(SUMX(ISTATE,IPT)**2)/NSAMPLS)/(NSAMPLS-1)
            STDEV(IPT)=SQRT(AMAX1(0.,VAR))
   60    CONTINUE
```

```
          WRITE(6,1020) ISTATE,(STDEV(IPT),IPT=1,11)
   70 CONTINUE
C
      WRITE(6,1040)
      DO 90 ISTATE=1,4
         DO 80 IPNT=1,11
            PFDIAG(IPNT)=SUMPFD(ISTATE,IPNT)/NSAMPLS
   80    CONTINUE
         WRITE(6,1020)ISTATE,(PFDIAG(IPNT),IPNT=1,11)
   90 CONTINUE
      WRITE (6,1050)
      RETURN
C
      ENTRY ESTIXO
C
C  INITIALIZE AT BEGINNING OF EACH RUN THROUGH PROGRAM
      IF (IPASS.EQ.1) RETURN
      IPNT=0
      NSAMPLS=NSAMPLS+1
C
C  INITIALIZE AT BEGINNING OF PROBLEM
      IF (IRUN.NE.1) GO TO 10
      DO 110 KK=1,44
         SUMX(KK)=0.0
         SUMX2(KK)=0.0
         SUMPFD(KK)=0.0
  110 CONTINUE
      GO TO 10
C
 1000 FORMAT(//T5,62("* ")//T5"STATISTICS AFTER "I3" RUNS OF CIRCULAR OR
     &BIT"   )
 1010 FORMAT(//T65"TIME"/6X"STATE"T15,11G10.2//T45
     &        "AVERAGES OF TRUE ERROR AT T-"  )
 1020 FORMAT(7X,I2,5X,11(G10.3))
 1030 FORMAT(1H0,T40"STANDARD DEVIATIONS OF TRUE ERROR AT T-"  )
 1040 FORMAT(1H0,T20"EXPECTED STANDARD DEVIATION OF TRUE ERROR - AVERAGE
     & OF SQRT(PF(I,I)) AT T-"  )
 1050 FORMAT(/T5,62("* ")//)
      END
```

```
      SUBROUTINE FQGEN(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NZF,NZQ,F,QF)
C
C    USER-WRITTEN SUBROUTINE
C    NONLINEAR SATELLITE ORBIT PROBLEM
C    FQGEN CALCULATES THE NON ZERO ELEMENTS OF THE MATRICES F
C    AND QF FOR THE PROPAGATION OF THE COVARIANCE MATRIX PF
C
      COMMON/QF/QFIN(2)
      COMMON/GO/GO
      DIMENSION XF(NF),XS(NS),F(NZF),QF(NZQ),XTRAJ(NXTJ)
C
      F(2)=XF(4)**2+2.*GO/(XF(1)**3)
      F(3)=2.*XF(1)*XF(4)
      F(5)=2.*XF(4)*XF(2)/XF(1)**2
      F(6)=-2.*XF(4)/XF(1)
      F(7)=-2.*XF(2)/XF(1)
      RETURN
C
      ENTRY FQGEND
      F(1)=1.
      F(4)=1.
      QF(1)=QFIN(1)
      QF(2)=QFIN(2)
      RETURN
      END
```

```
      SUBROUTINE HRZ(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF,
     *               IMEAS,M,H,RF,ZRES)
C
C   USER-WRITTEN SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C
      COMMON/RF/RFIN(2)
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),H(NF),STD(2)
C
      CALL ZROIZE(1,NF,H)
      RF=RFIN(IMEAS)
      IF (IMEAS-1) 10,10,20
C
C   RANGE MEASUREMENT
   10 H(1)=1.
      ZRES=XS(1)-XF(1)+GAUSS(0.,STD(1))
      RETURN
C
C   ANGLE MEASUREMENT
   20 H(3)=1.
      ZRES=XS(3)-XF(3)+GAUSS(0.,STD(2))
      RETURN
C
      ENTRY HRZO
      STD(1)=SQRT(RFIN(1))
      STD(2)=SQRT(RFIN(2))
      RETURN
      END
```

```
      SUBROUTINE SNOYS(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
C
C   USER-WRITTEN SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C   THE TRUTH MODEL IS WITHOUT DRIVING NOISE
C
      RETURN
      ENTRY SNOYSO
      RETURN
      END
```

```
      SUBROUTINE TRAJO(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ)
C
C   USER-WRITTEN SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C   NO TRAJECTORY COMPUTATIONS REQUIRED BEYOND THOSE FOR XS AND XF
C
      RETURN
      END
```

```
      SUBROUTINE USRIN
C
C   USER-WRITTEN SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C   THIS ROUTINE READS AND PRINTS THE USER-WRITTEN CONSTANTS
C
      COMMON/QF/QFIN(2)
      COMMON/RF/RFIN(2)
      COMMON/GO/GO
      NAMELIST /IN/RFIN,QFIN,GO
C
      READ(5,IN)
      WRITE(6,IN)
      RETURN
      END
```

```
      SUBROUTINE XFDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,NTR,PF,XDOT)
C
C   USER-WRITTEN SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C   XFDOT COMPUTES THE FILTER DERIVATIVES
C           XFDOT=F(XF,T)
C
      COMMON/GO/GO
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),PF(NTR),XDOT(NF)
C
      XDOT(1)=XF(2)
      XDOT(2)=XF(1)*XF(4)*XF(4)-GO/(XF(1)*XF(1))
      XDOT(3)=XF(4)
      XDOT(4)=-2.*XF(4)*XF(2)/XF(1)
      RETURN
C
      ENTRY XFDOTO
      RETURN
      END
```

```
      SUBROUTINE XSDOT(IRUN,T,NF,NS,NXTJ,XF,XS,XTRAJ,XDOT)
C
C   USER SUPPLIES SUBROUTINE
C   NONLINEAR SATELLITE ORBIT PROBLEM
C   XSDOT COMPUTES THE SYSTEM DERIVATIVES
C      XSDOT=G(XS,T)
C
      COMMON/GO/GO
      DIMENSION XF(NF),XS(NS),XTRAJ(NXTJ),XDOT(NS)
C
      XDOT(1)=XS(2)
      XDOT(2)=XS(1)*XS(4)*XS(4)-GO/(XS(1)*XS(1))
      XDOT(3)=XS(4)
      XDOT(4)=-2.*XS(4)*XS(2)/XS(1)
      RETURN
C
      ENTRY XSDOTO
      RETURN
      END
```

RUN DATE AND TIME : 07/29/80 08.21.52.

```
*************************************************** S O F E   B E G I N S ***************************************************

                                              S O F E

                                A GENERALIZED MONTE CARLO SIMULATION FOR
                        THE DESIGN AND PERFORMANCE ANALYSIS OF MULTISENSOR SYSTEMS
                          WHERE DATA IS COMBINED USING KALMAN FILTER TECHNIQUES


PROBLEM TITLE  :  SATELLITE ORBIT DETERMINATION USING AN EXTENDED KALMAN FILTER


RUN DATE AND TIME  :   07/29/80  08.21.52.
```

```
                 COMPOSITE OF INPUT DATA AND DEFAULT VALUES IN THE 'PRDATA' NAMELIST GROUP
-----------------------------------------------------------------------------------------------------------
DIMENSIONS AND SIZES:
     NF =   4       NS =   4       N =   2       NZF =    7       NZQ =    2       NXTJ =   0
EXTERNAL TRAJECTORY:
     LXTJ = F

BEGIN AND END TIME OF EACH RUN:
     TO = 0.                 TF =   5.0000

TIME WHEN UPDATES MAY BEGIN:
     TMEASO =  0.

TIME INTERVAL BETWEEN UPDATES, PRINTS, STORAGE FOR CALCOMP AND PRINTER PLOTS, CALLS TO ESTIX AND SNOYS:
     DTMEAS =     .50000          DTPRNT =    .10000E+10        DTCCPL =    .50000E-01
     DTPRPL =     .50000E-01      DTSTIX =    .50000            DTNOYS =    .10000E+10
PARAMETERS GOVERNING PRINTED OUTPUT:
     LPR =    T      LPRXS =  T      LPRXF =  T
     LPRZR =  T      LPRH =   F      LPRK =   F      LPRDG =  T      LPRLT =  T      LPRUD =  F
                                                     LPRXTJ = F      IPGSIZ = 55     IPRRUN =     1
PARAMETER GOVERNING DATA STORAGE FOR POSTRUN CALCOMP PLOTTING:
     LCC =  T

PARAMETERS GOVERNING PRINTER PLOTTING:
     LPP =  T      LPPLD =  F      LPPUP =  F

INTEGRATION CONTROL PARAMETERS:
     MODE =  1      TOLER =   .10000E-03     HMAX =   .10000E+10     HMIN =   .10000E-03     HO =   .10000E-01
CONTINUATION FLAG, RANDOM NUMBER SEED, NUMBER OF RUNS:
     ICONT =  0      ISEED =    23      IPASS =    50
-----------------------------------------------------------------------------------------------------------


THE FOLLOWING    7 ROW-COLUMN PAIRS LOCATE THE NONZERO ELEMENTS OF THE SPARSE MATRIX F
  1.      1   2         2.      2   1         3.      2   4         4.      3   4         5.      4   1         6.      4   2
  7.      4   4


THE FOLLOWING    2 ROW-COLUMN PAIRS LOCATE THE NONZERO ELEMENTS OF THE SPARSE MATRIX QF
  1.      2   2         2.      4   4
```

STRUCTURE OF UNLABELED COMMON AREA 'A'

VECTOR OR MATRIX	LENGTH	FIRST WORD LOCATION
GENERAL WORKING SPACE	NF+2*M	1
COMPOSITE OF (XS,XF,PF), Y	NALL	9
SYSTEM STATE VECTOR, XS	NS	9
FILTER STATE VECTOR, XF	NF	13
FILTER COVARIANCE, PF	NTR	17
SQUARE ROOT OF PF, S	NTR	27
INDICES OF NONZEROES IN F	2*NZF	37
INDICES OF NONZEROES IN QF	2*NZQ	51
VALUES OF NONZEROES IN F	NZF	55
VALUES OF NONZEROES IN QF	NZQ	62
INITIAL CONDITIONS FOR XS	NS	64
INITIAL CONDITIONS FOR XF	NF	68
INITIAL CONDITIONS FOR PF	NTR	72
DERIVATIVE OF Y	NALL	82
DERIVATIVE OF XS	NS	82
DERIVATIVE OF XF	NF	86
DERIVATIVE OF PF	NTR	90
D.E. WORKING SPACE	4*NALL	100
MEASUREMENT SENSITIVITY, H	NF	172
PRODUCT OF S AND H	NF	176
KALMAN GAIN	NF	180
XTRAJ TIMES	3	184
XTRAJ DATA	3*NXTJ	187
XTRAJ INTERPOLATED DATA	4*NXTJ	187
XTRAJ INTERPOLATION COEFFICIENTS	6*NXTJ	187

NOTE: NS	= DIMENSION OF SYSTEM STATE	=	4
NF	= DIMENSION OF FILTER STATE	=	4
M	= NUMBER OF MEASUREMENTS PER UPDATE	=	2
NTR	= SIZE OF TRIANGULAR PART OF PF	=	10
NALL	= DIMENSION OF COMPOSITE STATE Y (NS+NF+NTR)	=	18
NZF	= NUMBER OF NONZEROES IN F	=	7
NZQ	= NUMBER OF NONZEROES IN QF	=	2
NXTJ	= NUMBER OF VARIABLES ON XTRAJ TAPE	=	0

AVAILABLE SPACE IN BLANK COMMON 'A'   1000
SPACE IN 'A' REQUIRED BY USER'S PROBLEM   186

```
$IN
RFIN     = .1E-01, .4E-01,
QFIN     = .2E-01, .2E-01,
GO       = .1E+01,
$END
```

```
SATELLITE ORBIT DETERMINATION USING AN EXTENDED KALMAN FILTER               07/29/80    08.21.52.  RUN NUMBER -    1

STATE VECTOR XSO   AT T =  0.
    1.   1.00000          2.  0.                    3.  0.                    4.   1.00000
STATE VECTOR XFO   AT T =  0.
    1.   1.00000          2.  0.                    3.  0.                    4.   1.00000
SIGMAS  FOR  XFO   AT T =  0.
    1.   .316228          2.   .316228              3.   .316228              4.   .316228
COVARIANCE   PFO   AT T =  0.
   .100
  0.          .100
  0.         0.          .100
  0.         0.         0.          .100
MEASUREMENT   1   AT T =   .50000    ***  RESIDUAL VALUE =   .1884693       ***  RESIDUAL STD DEV =   .4762680
MEASUREMENT   2   AT T =   .50000    ***  RESIDUAL VALUE =   .5179488E-01   ***  RESIDUAL STD DEV =   .4053022
MEASUREMENT   1   AT T =  1.0000     ***  RESIDUAL VALUE =  -.2971662       ***  RESIDUAL STD DEV =   .2462051
MEASUREMENT   2   AT T =  1.0000     ***  RESIDUAL VALUE =   .2413848       ***  RESIDUAL STD DEV =   .3047464
MEASUREMENT   1   AT T =  1.5000     ***  RESIDUAL VALUE =  -.1589993       ***  RESIDUAL STD DEV =   .2410443
MEASUREMENT   2   AT T =  1.5000     ***  RESIDUAL VALUE =  -.1577056E-01   ***  RESIDUAL STD DEV =   .2763486
MEASUREMENT   1   AT T =  2.0000     ***  RESIDUAL VALUE =   .2616353       ***  RESIDUAL STD DEV =   .2295052
MEASUREMENT   2   AT T =  2.0000     ***  RESIDUAL VALUE =   .2078209       ***  RESIDUAL STD DEV =   .2711744
MEASUREMENT   1   AT T =  2.5000     ***  RESIDUAL VALUE =  -.1604872       ***  RESIDUAL STD DEV =   .1992877
MEASUREMENT   2   AT T =  2.5000     ***  RESIDUAL VALUE =  -.3961451       ***  RESIDUAL STD DEV =   .2859454
MEASUREMENT   1   AT T =  3.0000     ***  RESIDUAL VALUE =   .3338299       ***  RESIDUAL STD DEV =   .2120590
MEASUREMENT   2   AT T =  3.0000     ***  RESIDUAL VALUE =   .1255264       ***  RESIDUAL STD DEV =   .2879902
MEASUREMENT   1   AT T =  3.5000     ***  RESIDUAL VALUE =  -.9889211E-01   ***  RESIDUAL STD DEV =   .2051742
MEASUREMENT   2   AT T =  3.5000     ***  RESIDUAL VALUE =   .3458027       ***  RESIDUAL STD DEV =   .3038755
MEASUREMENT   1   AT T =  4.0000     ***  RESIDUAL VALUE =   .3554879       ***  RESIDUAL STD DEV =   .2168668
MEASUREMENT   2   AT T =  4.0000     ***  RESIDUAL VALUE =  -.1658885E-02   ***  RESIDUAL STD DEV =   .2949689
MEASUREMENT   1   AT T =  4.5000     ***  RESIDUAL VALUE =  -.2973854       ***  RESIDUAL STD DEV =   .2038792
MEASUREMENT   2   AT T =  4.5000     ***  RESIDUAL VALUE =  -.7366193E-02   ***  RESIDUAL STD DEV =   .2861469
MEASUREMENT   1   AT T =  5.0000     ***  RESIDUAL VALUE =   .5119759E-01   ***  RESIDUAL STD DEV =   .2100342
MEASUREMENT   2   AT T =  5.0000     ***  RESIDUAL VALUE =   .1493886       ***  RESIDUAL STD DEV =   .2770043

STATE VECTOR XS    AT T =  5.0000
    1.   1.00000          2.  0.                    3.   5.00000              4.   1.00000
STATE VECTOR XF    AT T =  5.0000
    1.   .896292          2.  -.243298              3.   4.92401              4.   1.04933
SIGMAS  FOR  XF    AT T =  5.0000
    1.  8.791316E-02      2.   .189713              3.   .138376              4.   .213268
COVARIANCE    PF   AT T =  5.0000
   7.729E-03
   1.206E-02   3.599E-02
  -3.043E-04   1.073E-02   1.915E-02
  -1.251E-02  -7.149E-03   1.206E-02   4.548E-02

RUN NUMBER      1 COMPLETE AT T =   5.000000

RUN NUMBER      2 COMPLETE AT T =   5.000000

RUN NUMBER      3 COMPLETE AT T =   5.000000

RUN NUMBER      4 COMPLETE AT T =   5.000000
```

```
RUN NUMBER     5 COMPLETE AT T =    5.000000
RUN NUMBER     6 COMPLETE AT T =    5.000000
RUN NUMBER     7 COMPLETE AT T =    5.000000
RUN NUMBER     8 COMPLETE AT T =    5.000000
RUN NUMBER     9 COMPLETE AT T =    5.000000
RUN NUMBER    10 COMPLETE AT T =    5.000000
RUN NUMBER    11 COMPLETE AT T =    5.000000
RUN NUMBER    12 COMPLETE AT T =    5.000000
RUN NUMBER    13 COMPLETE AT T =    5.000000
RUN NUMBER    14 COMPLETE AT T =    5.000000
RUN NUMBER    15 COMPLETE AT T =    5.000000
RUN NUMBER    16 COMPLETE AT T =    5.000000
RUN NUMBER    17 COMPLETE AT T =    5.000000
RUN NUMBER    18 COMPLETE AT T =    5.000000
RUN NUMBER    19 COMPLETE AT T =    5.000000
RUN NUMBER    20 COMPLETE AT T =    5.000000
RUN NUMBER    21 COMPLETE AT T =    5.000000
RUN NUMBER    22 COMPLETE AT T =    5.000000
RUN NUMBER    23 COMPLETE AT T =    5.000000
RUN NUMBER    24 COMPLETE AT T =    5.000000
RUN NUMBER    25 COMPLETE AT T =    5.000000
RUN NUMBER    26 COMPLETE AT T =    5.000000
RUN NUMBER    27 COMPLETE AT T =    5.000000
RUN NUMBER    28 COMPLETE AT T =    5.000000
RUN NUMBER    29 COMPLETE AT T =    5.000000
RUN NUMBER    30 COMPLETE AT T =    5.000000
RUN NUMBER    31 COMPLETE AT T =    5.000000
```

```
RUN NUMBER    32 COMPLETE AT T =   5.000000
RUN NUMBER    33 COMPLETE AT T =   5.000000
RUN NUMBER    34 COMPLETE AT T =   5.000000
RUN NUMBER    35 COMPLETE AT T =   5.000000
RUN NUMBER    36 COMPLETE AT T =   5.000000
RUN NUMBER    37 COMPLETE AT T =   5.000000
RUN NUMBER    38 COMPLETE AT T =   5.000000
RUN NUMBER    39 COMPLETE AT T =   5.000000
RUN NUMBER    40 COMPLETE AT T =   5.000000
RUN NUMBER    41 COMPLETE AT T =   5.000000
RUN NUMBER    42 COMPLETE AT T =   5.000000
RUN NUMBER    43 COMPLETE AT T =   5.000000
RUN NUMBER    44 COMPLETE AT T =   5.000000
RUN NUMBER    45 COMPLETE AT T =   5.000000
RUN NUMBER    46 COMPLETE AT T =   5.000000
RUN NUMBER    47 COMPLETE AT T =   5.000000
RUN NUMBER    48 COMPLETE AT T =   5.000000
RUN NUMBER    49 COMPLETE AT T =   5.000000
```

* *

STATISTICS AFTER 50 RUNS OF CIRCULAR ORBIT

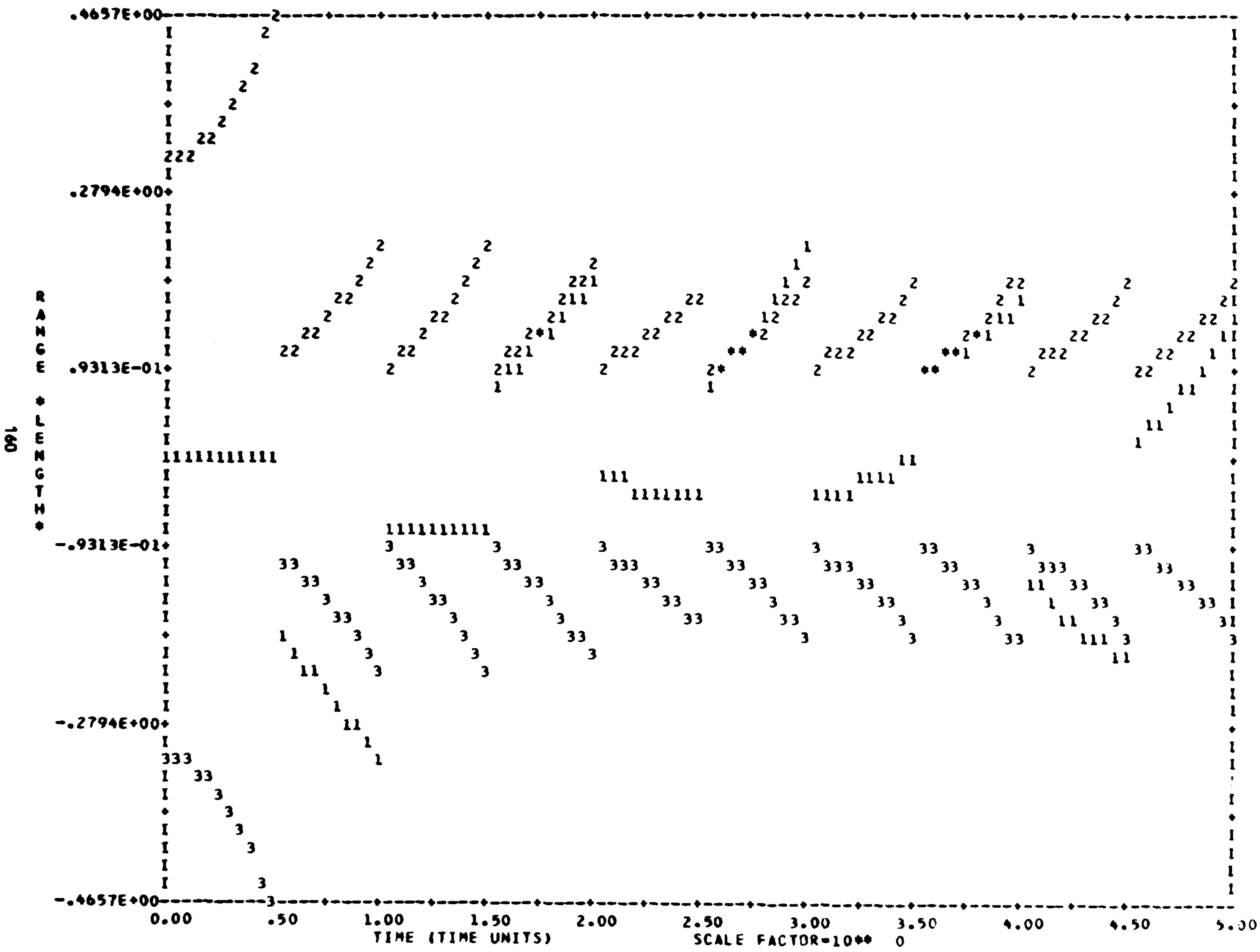
STATE	0.	.50	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
						TIME					
				AVERAGES OF TRUE ERROR AT T-							
1	0.	0.	-.516E-01	.832E-02	.326E-01	.282E-01	.441E-01	.202E-01	.193E-01	.686E-01	.135E-01
2	0.	0.	-.471E-01	.605E-01	.762E-01	.881E-01	.114	.850E-01	.948E-01	.156	-.277E-01
3	0.	0.	.571E-02	.380E-01	.128E-01	.419E-01	.442E-01	.468E-01	.669E-01	.634E-01	.438E-02
4	0.	0.	.274E-01	-.275E-01	-.611E-01	.122E-02	-.417E-01	-.805E-02	-.239E-02	-.796E-01	.259
			STANDARD DEVIATIONS OF TRUE ERROR AT T-								
1	0.	0.	.154	.155	.163	.150	.140	.142	.158	.158	.842E-01
2	0.	0.	.178	.205	.212	.231	.186	.176	.204	.249	.319
3	0.	0.	.182	.179	.177	.208	.189	.171	.158	.214	.132
4	0.	0.	.190	.289	.289	.299	.270	.321	.337	.383	1.02
		EXPECTED STANDARD DEVIATION OF TRUE ERROR = AVERAGE OF SQRT(PF(I,I)) AT T-									
1	.316	.466	.234	.223	.202	.187	.180	.179	.180	.181	.870E-01
2	.316	.623	.448	.401	.339	.306	.299	.297	.301	.306	.198
3	.316	.355	.239	.216	.207	.205	.206	.204	.202	.208	.141
4	.316	.427	.349	.375	.389	.365	.362	.352	.352	.419	.277

* *

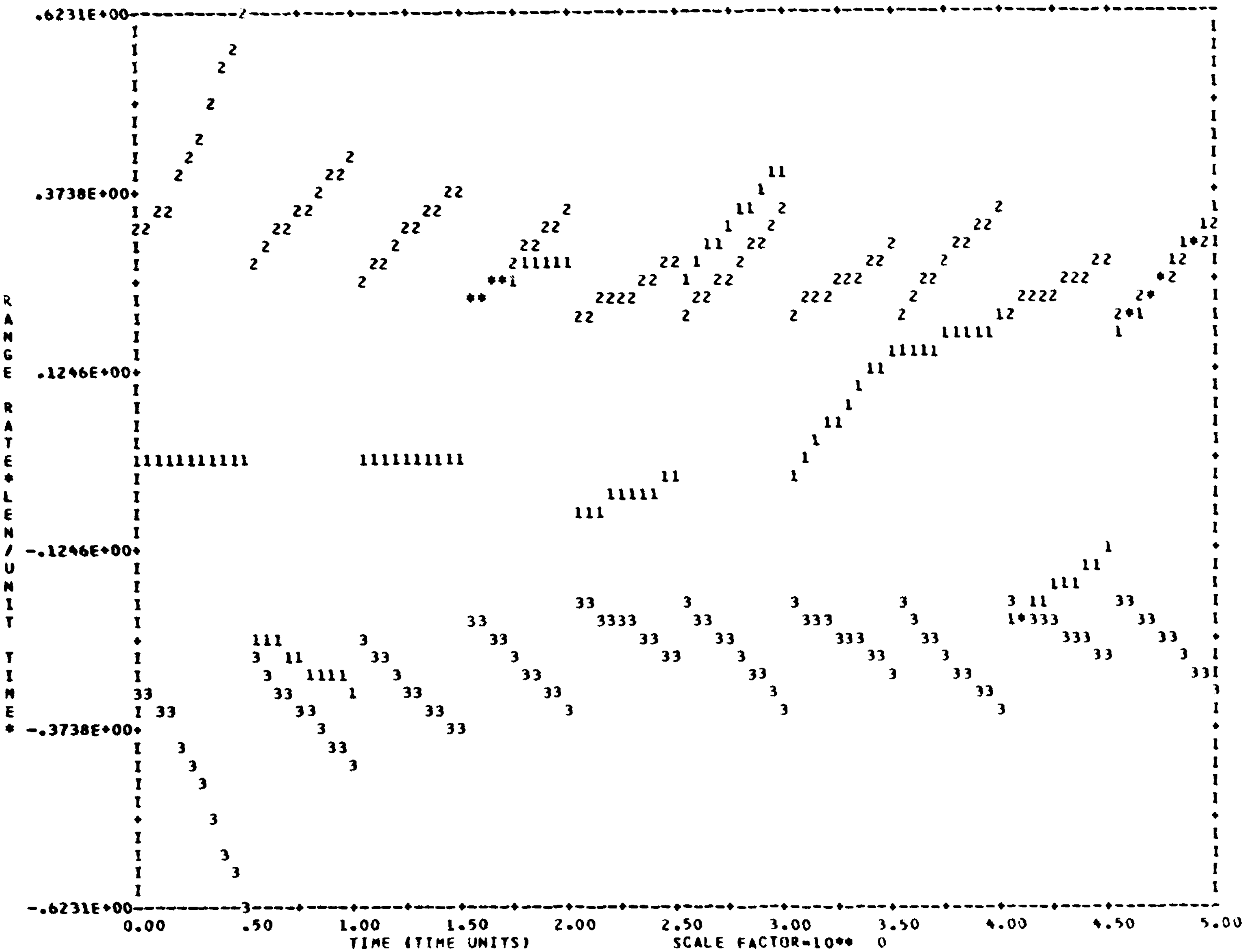
RUN NUMBER 50 COMPLETE AT T = 5.000000

KALMAN FILTER SIMULATION COMPLETE AFTER RUN 50

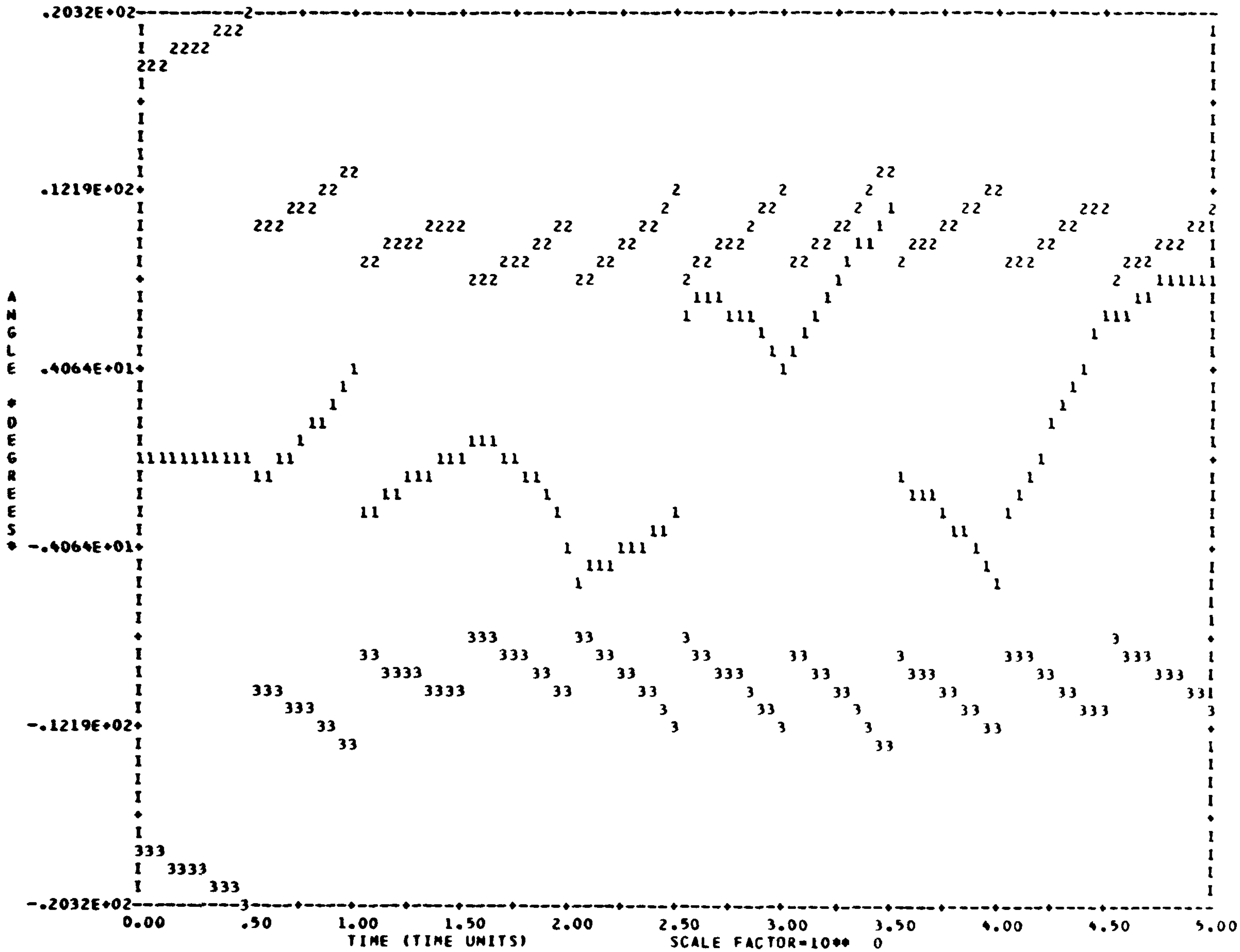
TRUE RANGE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3



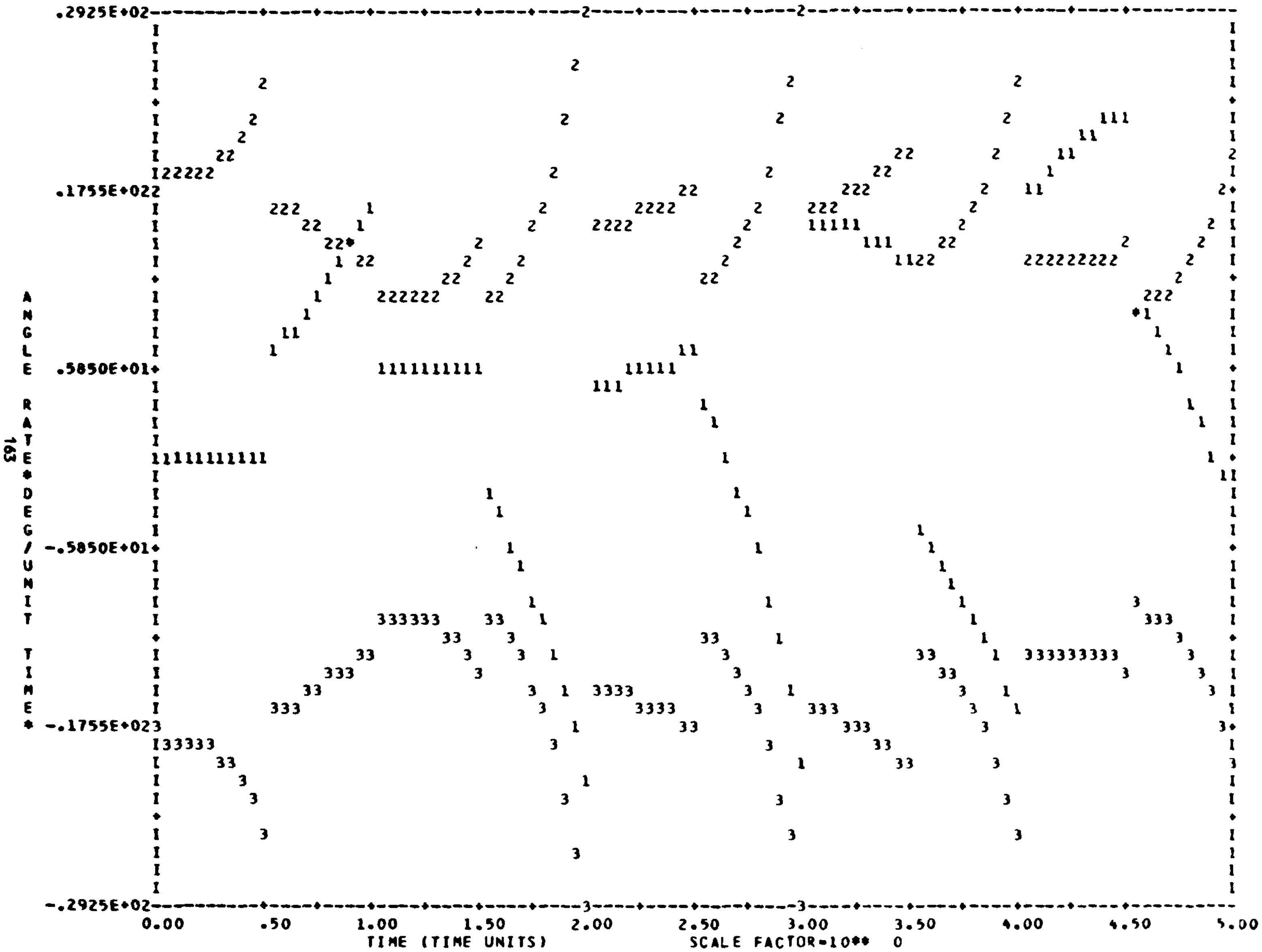
TRUE RANGE RATE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3



TRUE ANGLE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3



TRUE ANGLE RATE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3



```
FINAL CHECK PRODUCT IS  -.617886282939237E-20

****************************************************** S O F E   F I N I S H E D ******************************************************
```

```
  CSB    NOS/BE L499D      L499D-CMR3 06/16/80
08.16.34.SHMGK25  FROM CSA/GK
08.16.35.IP  00001216 WORDS - FILE INPUT  , DC 04
08.16.35.SHM,T60,IO100,CM70000. V720130, MUSICK
08.16.39.
08.16.39.* 'SOFE' EXAMPLE.
08.16.39.* EXTENDED KALMAN FILTER.
08.16.39.* NONLINEAR SATELLITE ORBIT PROBLEM.
08.16.39.
08.16.39.* SINGLE DATA CARD FOR FOLLOWING UPDATE
08.16.39.*  READS "*C SOFE.ZROIZE".
08.16.39.ATTACH,OLDPL,SOFE,CY=999,ID=SHM,SN=AFAL,
08.16.39.MR=1.
08.16.41.UPDATE,Q,C=COMPILE.
08.16.49. UPDATE COMPLETE.
08.16.50.
08.16.50.* COMPILE BASIC SOFE ROUTINES.
08.16.50.FTN,I=COMPILE,L=0.
08.20.16.      9.874 CP SECONDS COMPILATION TIME
08.20.16.RETURN,OLDPL,COMPILE.
08.20.17.
08.20.17.* COMPILE USER-WRITTEN SUBROUTINES.
08.20.17.FTN,I=INPUT,R=0,P.
08.20.37.       .995 CP SECONDS COMPILATION TIME
08.20.37.
08.20.37.* RUN SOFE USING TAPE5 DATA ON INPUT FIL
08.20.37.E.
08.20.37.REQUEST,TAPE4,*PF.
08.20.38.LGO(INPUT,TAPE3,TAPE9,OUTPUT,TAPE4,OUTPU
08.20.38.T,TAPE8,TAPE10,TAPE7)
08.25.35.     STOP  SOFE DONE
08.25.35.     37.488 CP SECONDS EXECUTION TIME
08.25.35.
08.25.35.* SAVE TAPE4 FOR LATER PLOTTING.
08.25.35.CATALOG,TAPE4,SOFEORBITPLOTTAPE,ID=SHM.
08.25.36.INITIAL CATALOG
08.25.36.RP = 008 DAYS
08.25.36.CT ID=      SHM PFN=SOFEORBITPLOTTAPE
08.25.36.CT CY= 001  00102528 WORDS.
08.25.37.OP  00006144 WORDS - FILE OUTPUT , DC 40
08.25.37.MS  111104  WORDS  (  129024 MAX USED)
08.25.37.CPA      51.099 SEC.       25.584 ADJ.
08.25.37.IO       78.805 SEC.       44.604 ADJ.
08.25.37.CM     2647.681 KWS.       21.498 ADJ.
08.25.37.CRUS                       91.686
08.25.37.COST                 $      4.18
08.25.37.PP      112.567 SEC.     DATE 07/29/80
08.25.37.EJ  END OF JOB, GK  V720130.
```

Figure B-1.

RANGE ERROR, AVG. AND STD. DEV., CIRCULAR ORBIT

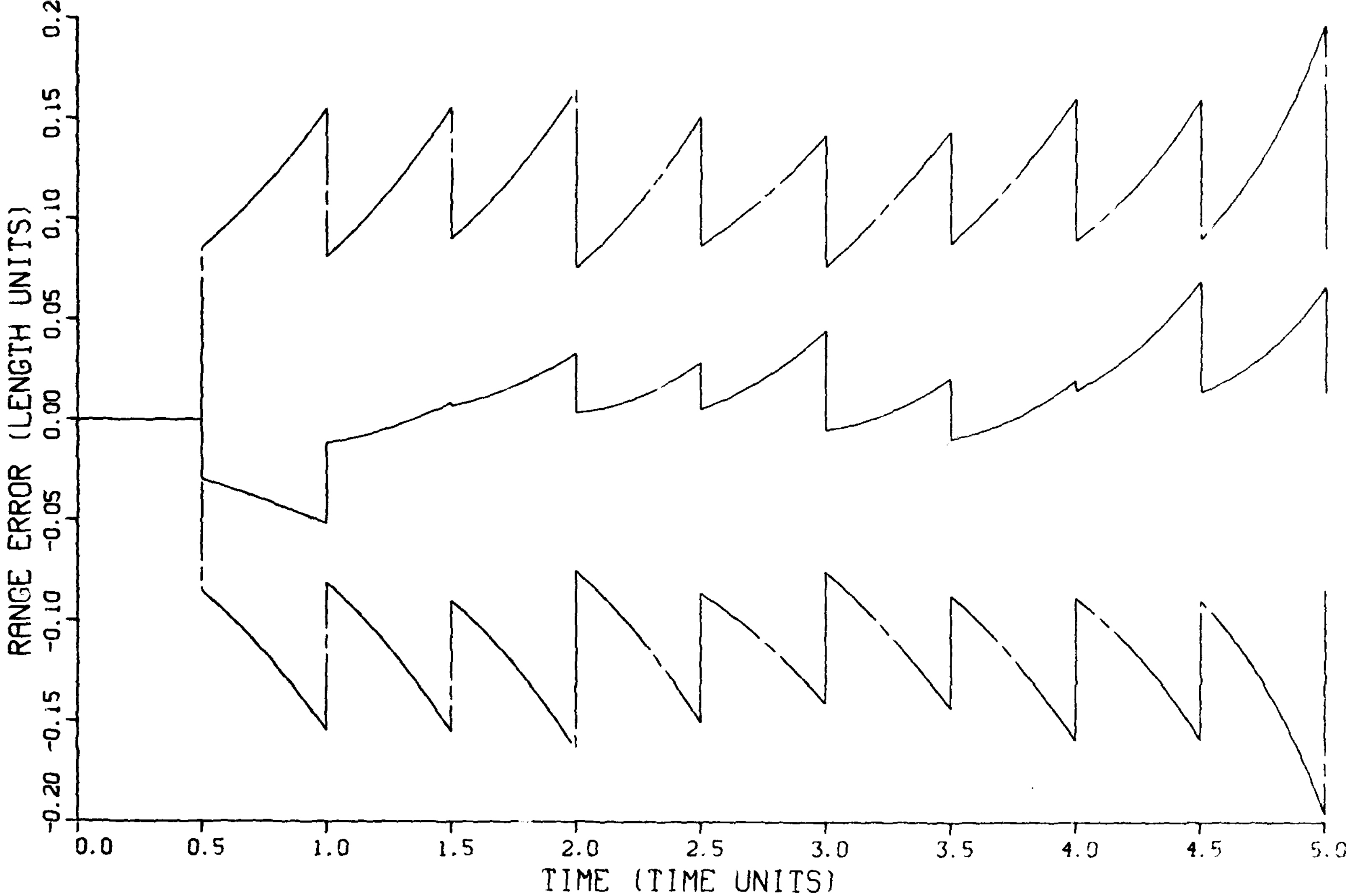


Figure B-2.
RANGE RATE ERROR, AVG. AND STD. DEV., CIRCULAR ORBIT

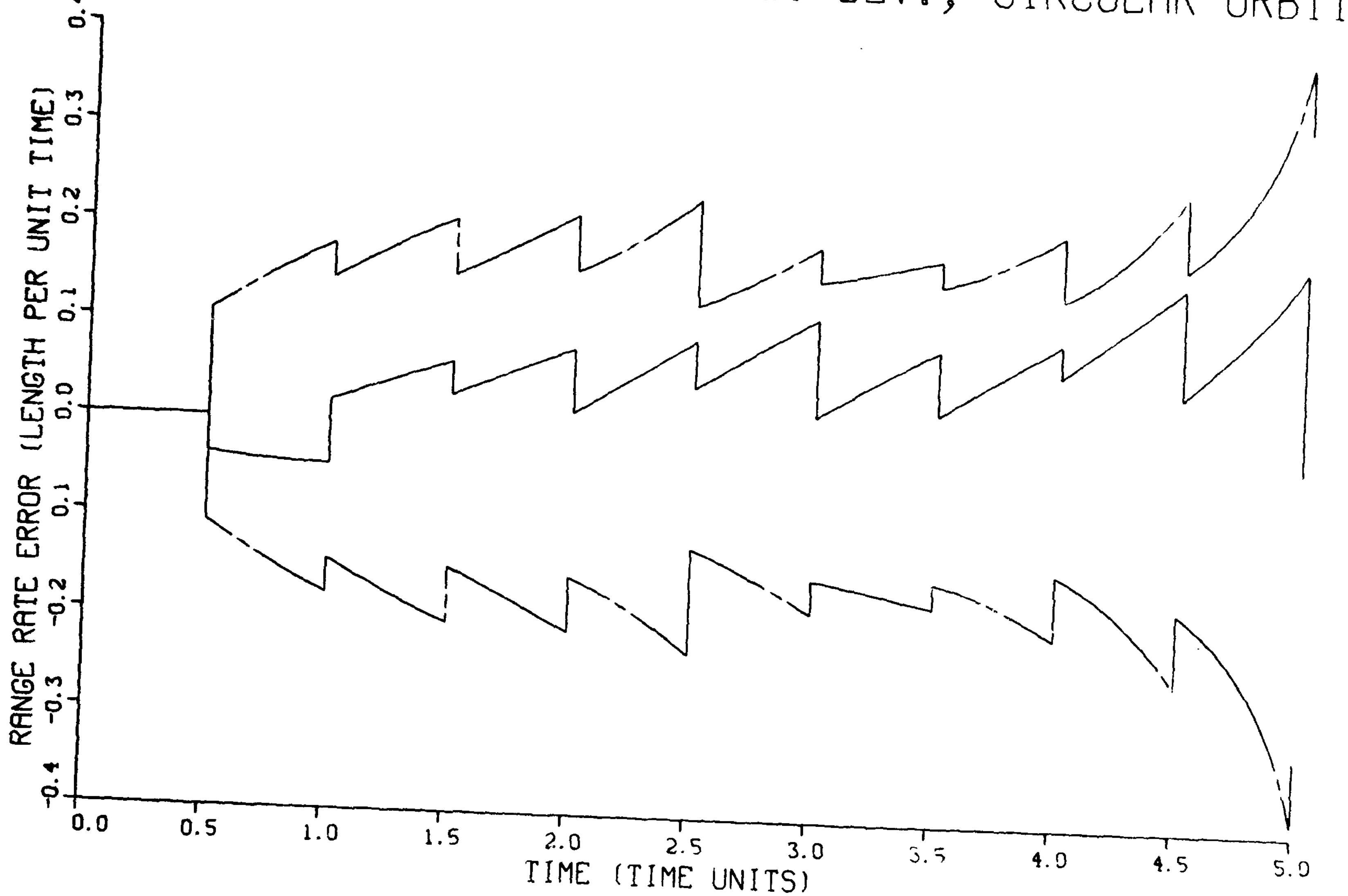


Figure B-3.

ANGLE ERROR, AVG. AND STD. DEV., CIRCULAR ORBIT

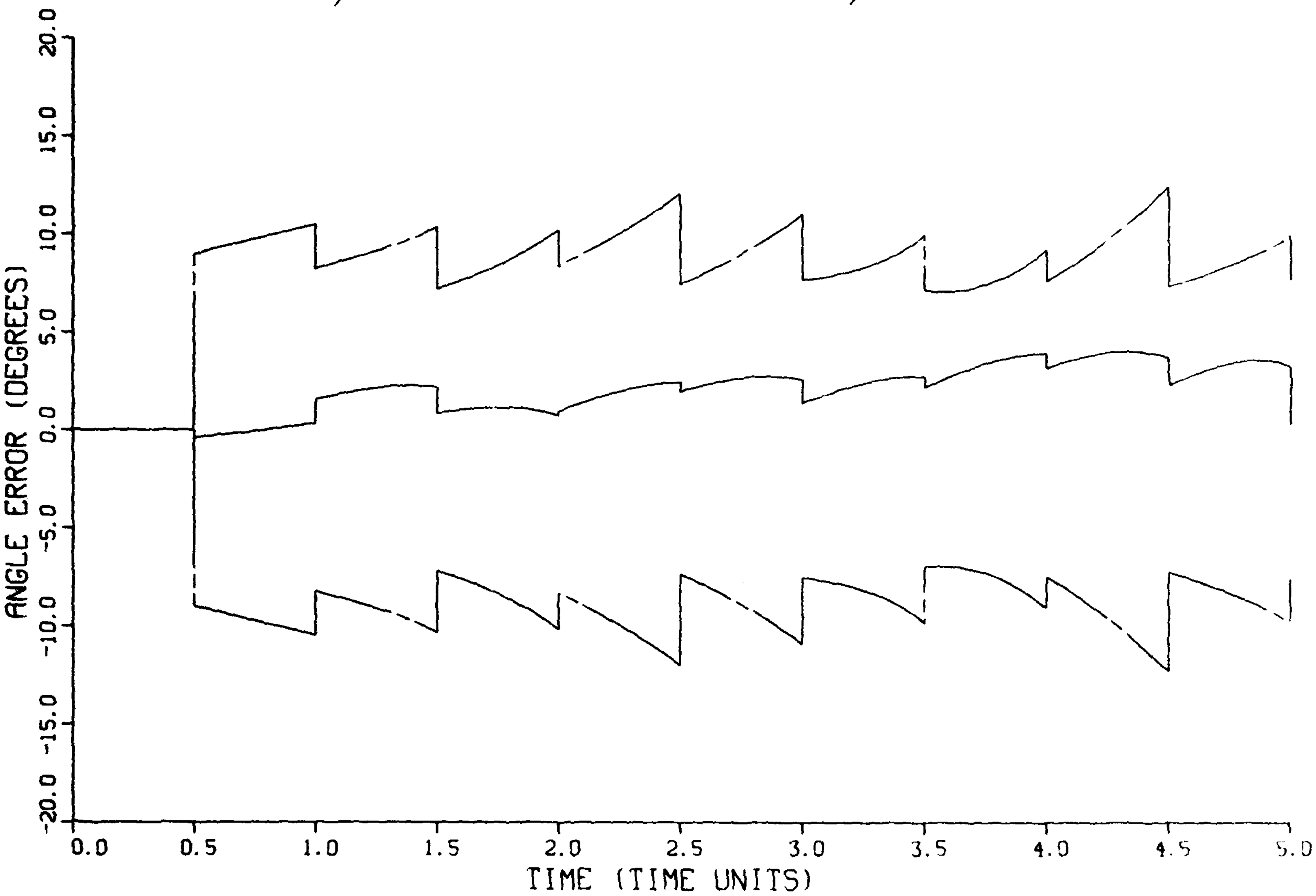


Figure B-4.
ANGLE RATE ERROR, AVG. AND STD. DEV., CIRCULAR ORBIT

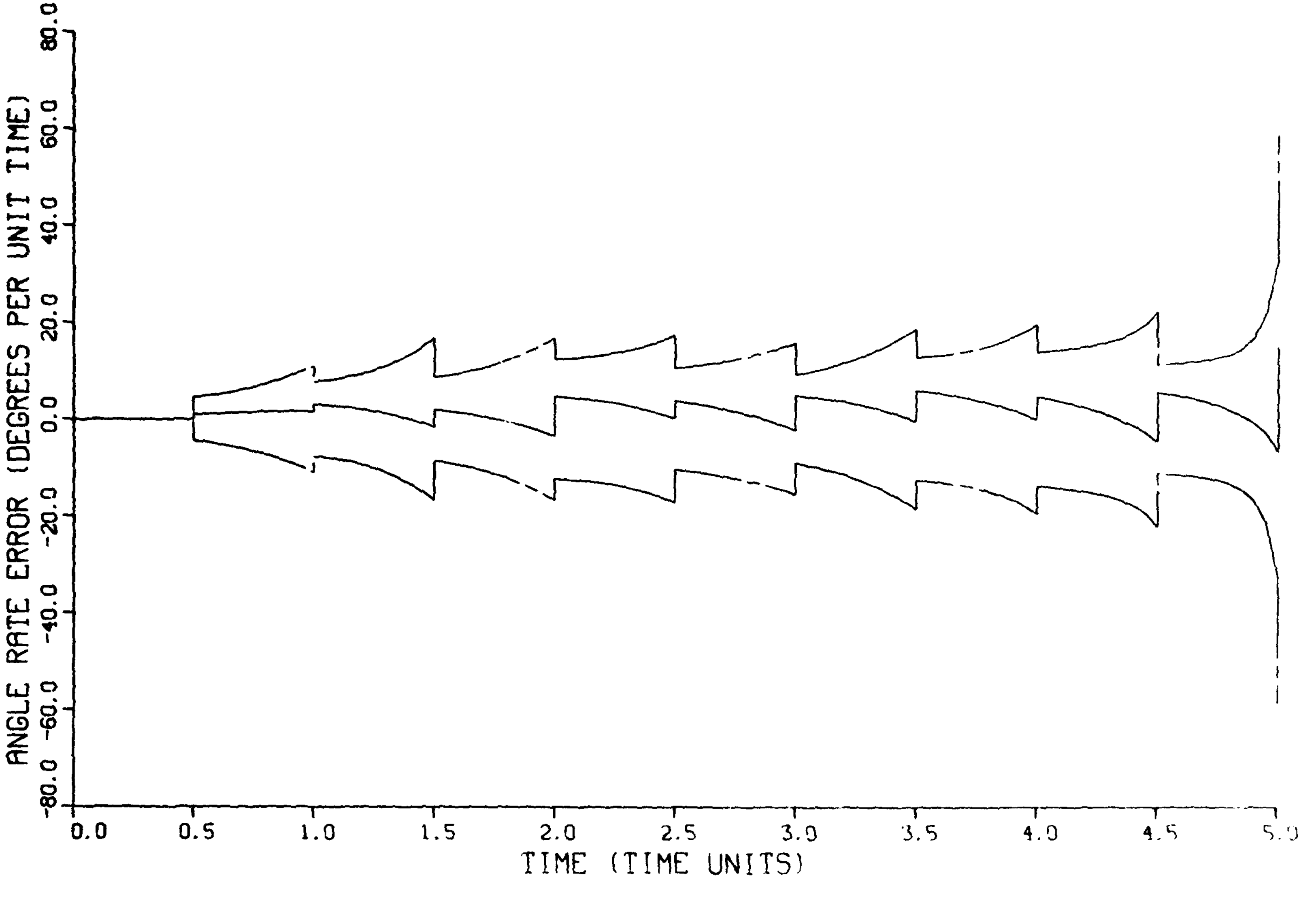


Figure B-5.

ACTUAL AND PRED. ESTIMATION ERROR, CIRC. ORBIT, RANGE

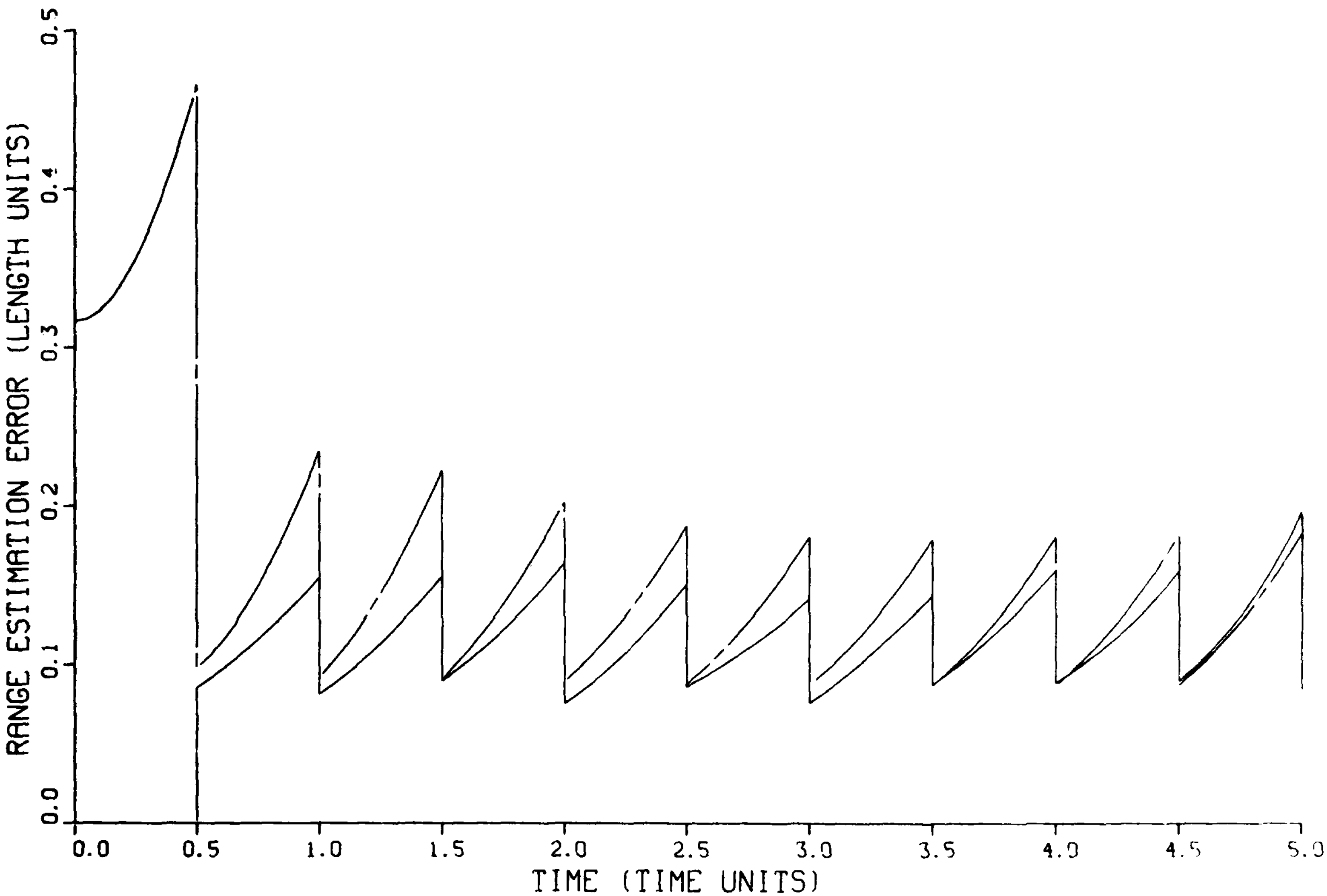


Figure B-6.
ACTUAL AND PRED. ESTIMATION ERROR, CIRC. ORBIT, RANGE RATE

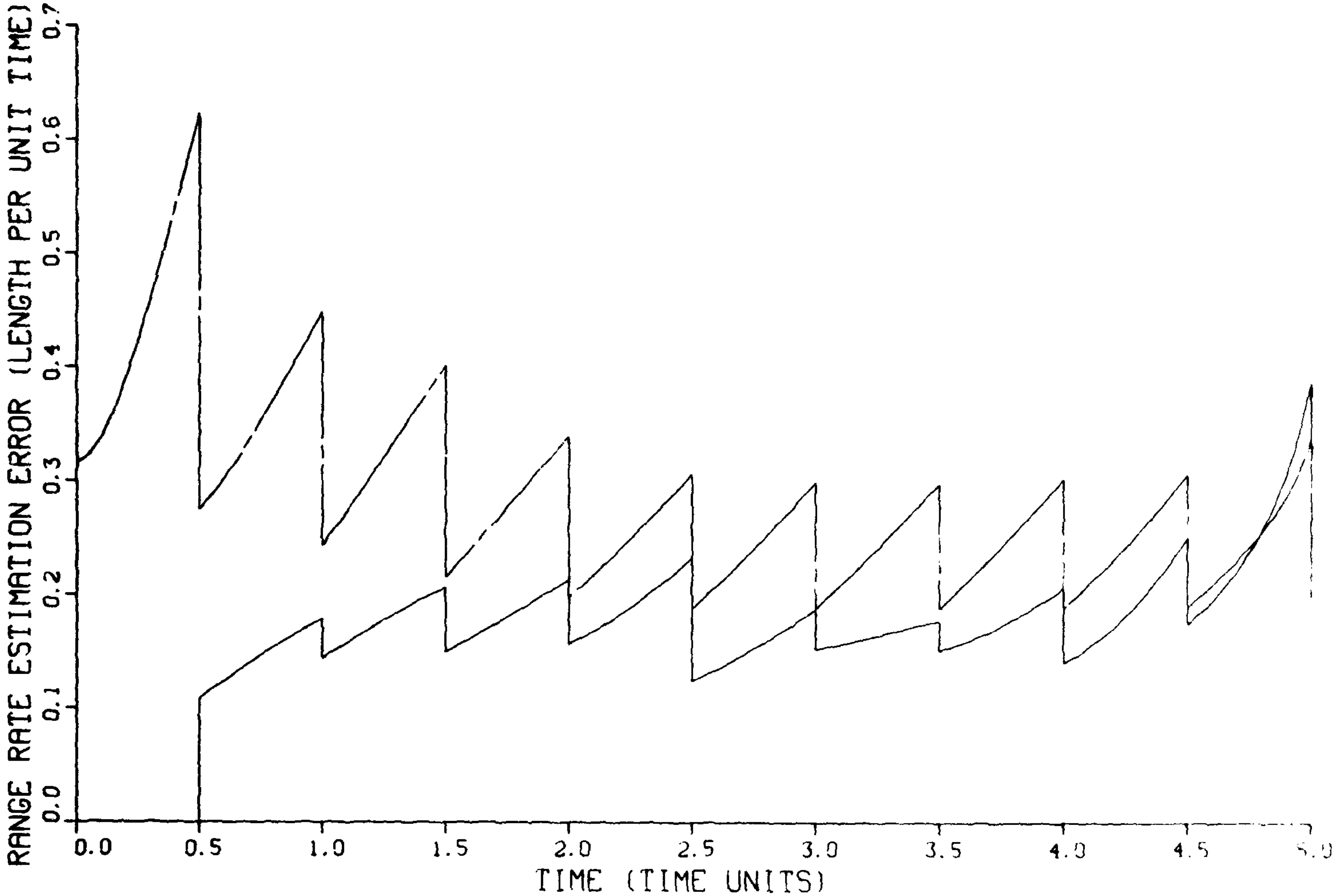
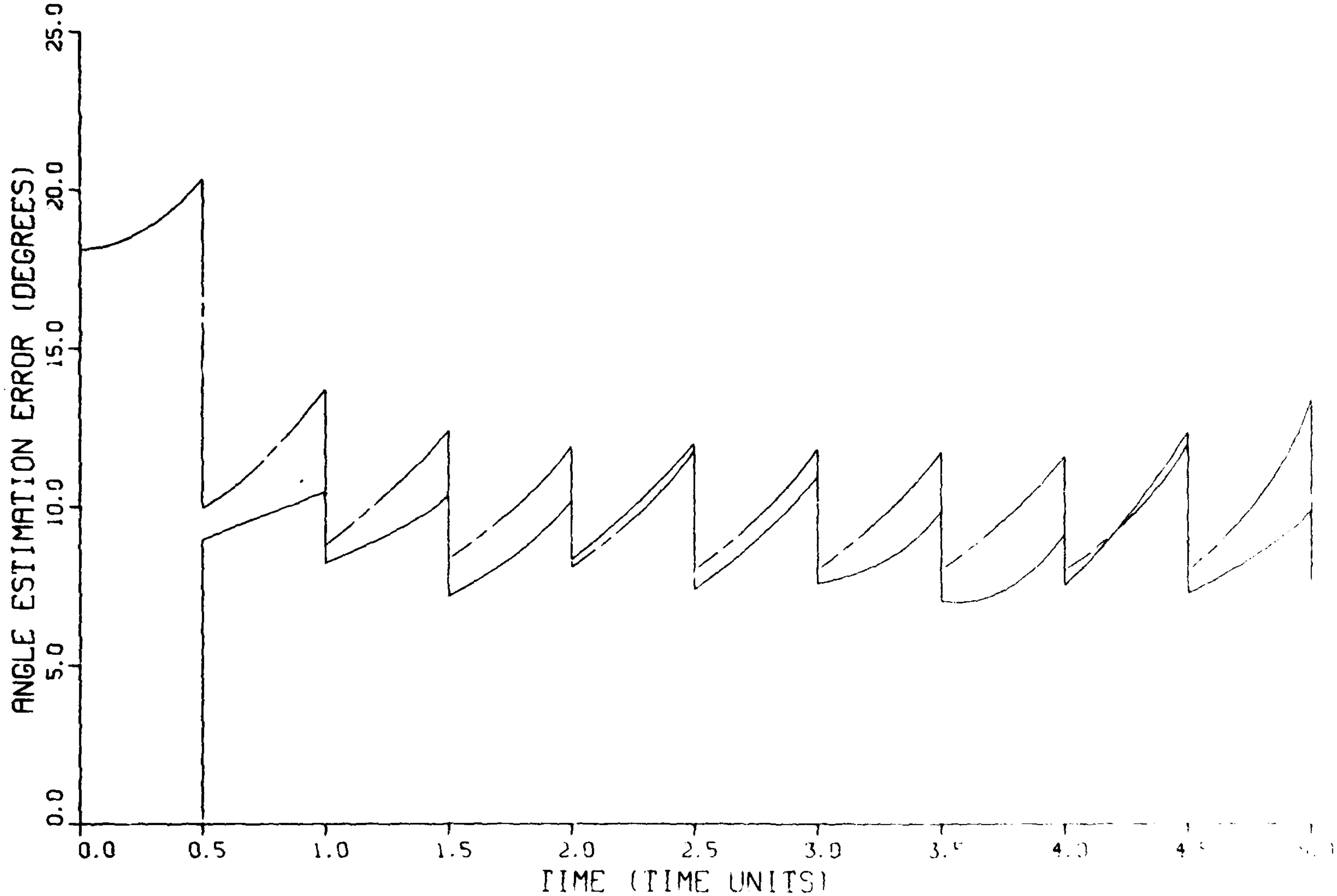
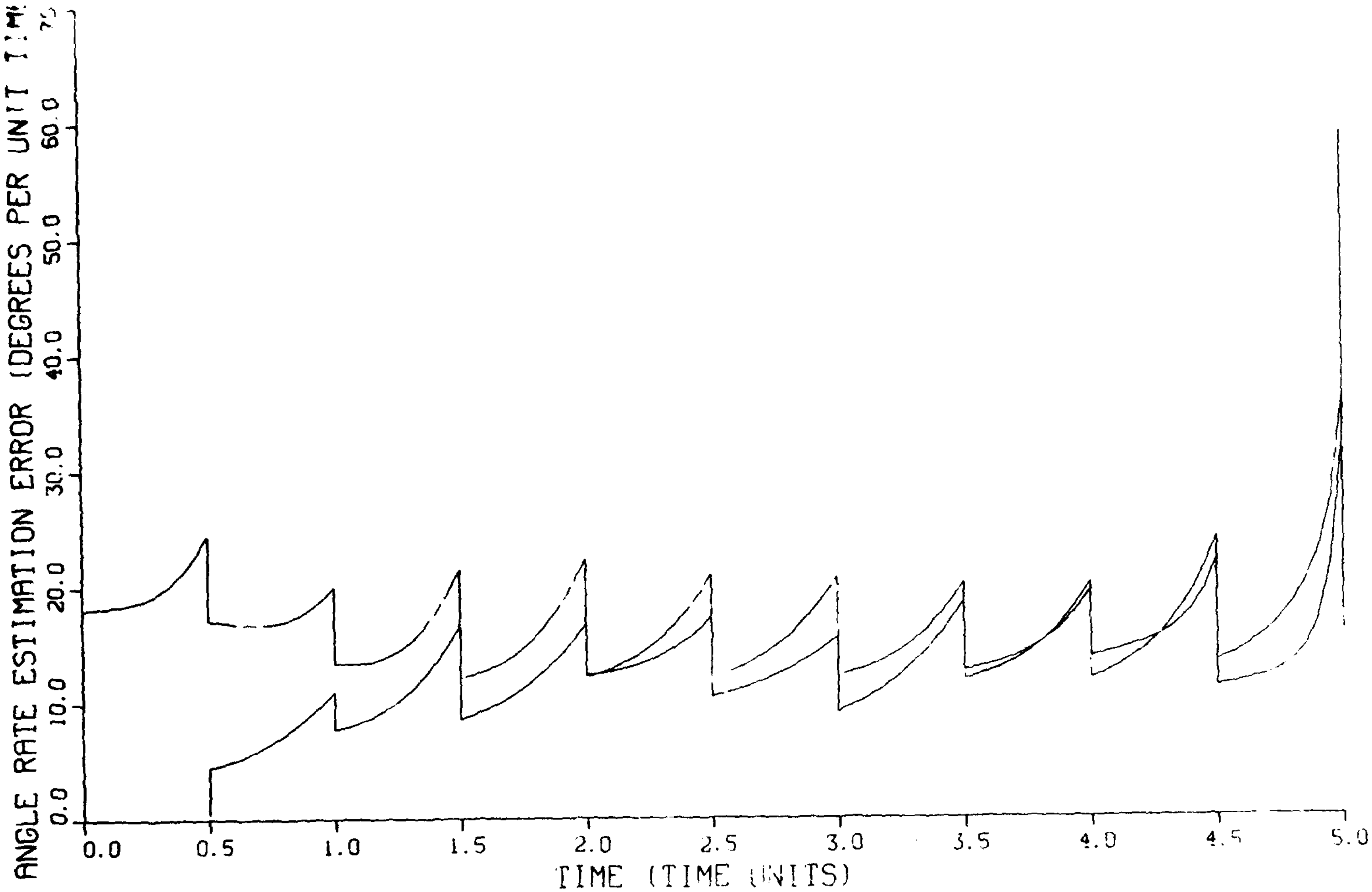


Figure B-7.

ACTUAL AND PRED. ESTIMATION ERROR, CIRC. ORBIT, ANGLE



-AND PRED. ESTIMATION ERROR, CIRC. ORBIT, ANGLE RATE



PRECEDING PAGE BLANK-NOT FILMED

APPENDIX C

Standard Short Test Output

```
SCAP/SCAP SINGLE AXIS INS --- A STANDARD SHORT TEST FOR "SOFE"            07/17/83    11.57.07.   RUN NUMBER -    1

STATE VECTOR XSO   AT T =  0.
    1.  0.                 2.  0.                 3.  0.                 4.  0.                 5.  0.
    6. -8.289704E-09       7.  0.                 8.  0.                 9.  0.
STATE VECTOR XFO   AT T =  0.
    1.  0.                 2.  0.                 3.  0.                 4.  0.                 5.  0.
SIGMAS  FOR  XFO   AT T =  0.
    1.   120.000           2.   2.00000           3.  1.745279E-03       4.  4.847680E-08       5.  6.439720E-03
COVARIANCE   PFO   AT T =  0.
  1.440E+04
  0.          4.00
  0.          0.          3.046E-05
  0.          0.          0.          2.350E-15
  0.          0.          0.          0.          4.147E-05

**********************************  BEGIN MEASUREMENT UPDATE AT T =   30.000     **********************************

STATE VECTOR XS-   AT T =   30.000
    1.  1.201097E-03       2.  1.201039E-04       3. -2.486337E-07       4. -5.305425E-10       5.  5.296677E-04
    5. -8.289704E-09       7.  1.940909E-04       8.  -79.7649           9.  5.182304E-02
STATE VECTOR XF-   AT T =   30.000
    1.  0.                 2.  0.                 3.  0.                 4.  0.                 5.  0.
SIGMAS  FOR  XF-   AT T =   30.000
    1.   120.000           2.   2.62138           3.  1.744072E-03       4.  4.847685E-08       5.  6.440134E-03
COVARIANCE   PF-   AT T =   30.000
  1.440E+04
  7.802E-06    6.87
 -2.102E-09  -2.934E-03   3.042E-05
 -1.619E-17  -3.386E-11   7.019E-14   2.350E-15
  8.355E-10   1.184E-03   8.355E-10   0.           4.148E-05
MEASUREMENT   1   AT T =   30.000     ***  RESIDUAL VALUE =   93.02448      ***  RESIDUAL STD DEV =   156.2050
MEAS. VECTOR  H    AT T =   30.000
    1.   1.00000           2.  0.                 3.  0.                 4.  0.                 5.  0.
KALMAN  GAIN  K    AT T =   30.000
    1.   .590164           2.  3.197351E-10       3. -8.614067E-14       4. -6.635247E-22       5.  3.424125E-14
MEASUREMENT   2   AT T =   30.000     ***  RESIDUAL VALUE =   .1528633      ***  RESIDUAL STD DEV =   2.668638
MEAS. VECTOR  H    AT T =   30.000
    1.  0.                 2.   1.00000           3.  0.                 4.  0.                 5.  0.
KALMAN  GAIN  K    AT T =   30.000
    1.  4.489633E-07       2.   .964896           3. -4.119781E-04       4. -4.754393E-12       5.  1.662162E-04

STATE VECTOR XF+   AT T =   30.000
    1.   54.8997           2.   .147497           3. -6.297633E-05       4. -7.267723E-13       5.  2.540837E-05
SIGMAS  FOR  XF+   AT T =   30.000
    1.   76.8221           2.   .491146           3.  1.353907E-03       4.  4.847685E-08       5.  5.424840E-03
COVARIANCE   PF+   AT T =   30.000
  5.902E+03
  1.122E-07    .241
  4.558E-10  -1.030E-04   1.833E-05
  9.566E-18  -1.189E-12   5.624E-14   2.350E-15
 -1.890E-10   4.155E-05   4.885E-07   5.628E-15   4.128E-05
```


PRECEDING PAGE BLANK-NOT FILMED

```
*********************************   END  MEASUREMENT UPDATE AT T =   30.000     *********************************


STATE VECTOR XS+C AT T =    30.000
    1.  -54.8985             2.  -.147377             3.  6.272770E-05         4.  -5.298158E-10        5.  5.042593E-04
    6.  -8.289704E-09        7.  1.940939E-04         8.  -79.7649             9.  5.182304E-02
STATE VECTOR XF+C AT T =    30.000
    1.  0.                   2.  0.                   3.  0.                   4.  0.                   5.  0.

*********************************   BEGIN MEASUREMENT UPDATE AT T =   60.000     *********************************


STATE VECTOR XS-  AT T =    60.000
    1.  -59.9197             2.  -.187538             3.  6.222293E-05         4.  2.561507E-09         5.  -2.812877E-03
    6.  -8.289704E-09        7.  8.971686E-05         8.  -132.798             9.   .102179
STATE VECTOR XF-  AT T =    60.000
    1.  0.                   2.  0.                   3.  0.                   4.  0.                   5.  0.
SIGMAS  FOR  XF-  AT T =    60.000
    1.   76.8221             2.   1.46997             3.  1.352869E-03         4.  4.847690E-08         5.  6.427954E-03
COVARIANCE   PF-  AT T =    60.000
  5.902E+03
  1.463E-05    2.16
 -9.511E-10  -1.858E-03   1.830E-06
 -4.793E-17  -8.874E-11   1.259E-13   2.350E-15
  4.084E-10   7.895E-04   4.426E-07   5.050E-15   4.132E-05
MEASUREMENT   1   AT T =    60.000       ***  RESIDUAL VALUE =  -217.4035      ***  RESIDUAL STD DEV =   126.1017
MEAS. VECTOR  H   AT T =    60.000
    1.   1.00000             2.  0.                   3.  0.                   4.  0.                   5.  0.
KALMAN  GAIN  K   AT T =    60.000
    1.   .371134             2.  9.198656E-11         3.  -5.981067E-14        4.  -3.014455E-21        5.  2.568093E-14
MEASUREMENT   2   AT T =    60.000       ***  RESIDUAL VALUE =   .1155597      ***  RESIDUAL STD DEV =   1.552680
MEAS. VECTOR  H   AT T =    60.000
    1.  0.                   2.   1.00000             3.  0.                   4.  0.                   5.  0.
KALMAN  GAIN  K   AT T =    60.000
    1.  3.815581E-07         2.   .896301             3.  -7.704940E-04        4.  -3.681079E-11        5.  3.274698E-04


STATE VECTOR XF+  AT T =    60.000
    1.  -80.6858             2.   .103576             3.  -8.903805E-05        4.  -4.253843E-12        5.  3.784231E-05
SIGMAS  FOR  XF+  AT T =    60.000
    1.   60.9208             2.   .473366             3.  6.317023E-04         4.  4.847687E-08         5.  6.407813E-03
COVARIANCE   PF+  AT T =    60.000
  3.711E+03
  9.539E-08    .224
  1.106E-10  -1.926E-04   3.990E-07
  3.716E-18  -9.203E-12   5.755E-14   2.350E-15
 -4.442E-11   8.187E-05   1.051E-05   3.411E-14   4.106E-05

*********************************   END  MEASUREMENT UPDATE AT T =   60.000     *********************************


STATE VECTOR XS+C AT T =    60.000
    1.   20.7661             2.  -.291115             3.  1.512610E-04         4.  2.565751E-09         5.  -2.850719E-03
    6.  -8.289704E-09        7.  8.971686E-05         8.  -132.798             9.   .102179
```

```
STATE VECTOR XF+C AT T =   60.000
    1.  0.                 2.  0.                 3.  0.                 4.  0.                 5.  0.

STATE VECTOR XS    AT T =   61.000
    1.   20.4712           2.  -.298741           3.  1.512411E-04       4.  2.565048E-09       5. -2.841232E-03
    6. -8.289704E-09       7.  8.969194E-05       8.  -132.724           9.   .101839
STATE VECTOR XF    AT T =   61.000
    1.  0.                 2.  0.                 3.  0.                 4.  0.                 5.  0.
SIGMAS  FOR  XF    AT T =   61.000
    1.   60.9208           2.   .486858           3.  6.316873E-04       4.  4.847687E-08       5.  6.408040E-03
COVARIANCE    PF   AT T =   61.000
  3.711E+03
  1.030E-07    .237
  1.012E-10  -2.044E-04   3.990E-07
  3.231E-18  -1.106E-11   5.988E-14   2.350E-15
 -4.019E-11   8.886E-05   1.047E-05   3.399E-14   4.106E-05

RUN NUMBER      1 COMPLETE AT T =   61.00000

KALMAN FILTER SIMULATION COMPLETE AFTER RUN      1

FINAL CHECK PRODUCT IS  -.132880246897869-143

**************************************************  S O F E    F I N I S H E D  **************************************************
```

APPENDIX D

Job Control

APPENDIX D

Job Control

SOFE was developed on the CDC CYBER-74 computer system at Wright-Patterson AFB, Ohio, using the NOS/BE operating system. The central memory requirements of SOFE are usually too large for it to run interactively on this system so various methods of batch-entry job control have been devised. These methods involve considerations of both program and data manipulation which in this case includes manipulation of: 1) basic SOFE; 2) user-written SOFE; and 3) the numerical data. Two possible methods for job control will be demonstrated here in Figures D-1 and D-2. Figure D-1 is the permanent file approach while D-2 is the card input approach.

Figure D-1 is the job control for the linear system example of Section 5.1. Figure D-1 illustrates job control setup and SOFE use when: 1) Basic SOFE and the user-written subroutines are on one local file named 'OLDPL' in the CDC UPDATE format; 2) the numerical data are on another local file named TAPE5. Both OLDPL and TAPE5 are permanently stored on disk, OLDPL being in a perm file named SOFE and TAPE5 being in a perm file named SOFEDATA. To create an object module, a full update is performed on the OLDPL file thereby producing a card-image file called COMPILE which is

then compiled. With this complete object module plus the TAPE5 data, SOFE execution can commence.

Figure D-2 is the job control for the nonlinear system example of Section 5.2. Figure D-2 illustrates deck setup and SOFE use when: 1) basic SOFE is part of an UPDATE file named OLDPL; 2) the user-written subroutines are on cards following the JCL; 3) the numerical data are on cards following the user-written subroutines. Selective update of basic SOFE is illustrated in the JCL using the '*C SOFE.ZROIZE' data card. Following compilation of basic SOFE, a compilation of the user-written subroutines occurs. Both object modules end up on the file named LGO.

The numerical input (problem title, PRDATA group, etc.) for the Figure D-2 example is at the end of the card deck on the INPUT file. Since SOFE expects its input on TAPE5, INPUT must be equated to TAPE5 at load time. This can be done under NOS/BE by inserting the name INPUT in the location reserved for TAPE5 on the LGO card, viz. the first location. For reference, all file assignments are shown on the LGO card in Figure D-2 even though only the first replacement is needed in this case.

```
SHM,T35,CM75000. V720130,MUSICK
COMMENT.*
COMMENT.* STANDARD LONG TEST FOR 'SOFE'
COMMENT.*
COMMENT.* ATTACH AND COMPILE BASIC SOFE WITH
COMMENT.  USER-WRITTEN ROUTINES APPENDED.
ATTACH,OLDPL,SOFE,CY=999,ID=SHM,SN=AFAL,MR=1.
UPDATE,F,C=COMPILE,O=OUTPUT.
FTN,I=COMPILE,L=0.
RETURN,OLDPL,COMPILE.
COMMENT.*
COMMENT.* ATTACH CARD INPUT DATA AND RUN SOFE.
ATTACH,TAPE5,SOFEDATA,CY=222,ID=SHM,SN=AFAL,MR=1.
LGO.
*EOR
```

Figure D-1. Job Control, All Files on Disk

```
SHM,T60,I0100,CM70000.  V720130, MUSICK
COMMENT.
COMMENT.*  'SOFE' EXAMPLE.
COMMENT.*  EXTENDED KALMAN FILTER.
COMMENT.*  NONLINEAR SATELLITE ORBIT PROBLEM.
COMMENT.
COMMENT.*  SINGLE DATA CARD FOR FOLLOWING UPDATE
COMMENT.*   READS "*C SOFE.ZROIZE".
ATTACH,OLDPL,SOFE,CY=999,ID=SHM,SN=AFAL,MR=1.
UPDATE,Q,C=COMPILE.
COMMENT.
COMMENT.*  COMPILE BASIC SOFE SUBROUTINES.
FTN,I=COMPILE,L=0.
RETURN,OLDPL,COMPILE.
COMMENT.
COMMENT.*  COMPILE USER-WRITTEN SUBROUTINES.
FTN,I=INPUT,R=0,P.
COMMENT.
COMMENT.*  RUN SOFE USING TAPE5 DATA ON INPUT FILE.
REQUEST,TAPE4,*PF.
LGO(INPUT,TAPE3,TAPE9,OUTPUT,TAPE4,OUTPUT,TAPE8,TAPE10,TAPE7).
COMMENT.
COMMENT.*  SAVE TAPE4 FOR LATER PLOTTING.
CATALOG,TAPE4,SOFEORBITPLOTTAPE,ID=SHM.
7/8/9
*C SOFE.ZROIZE
7/8/9
    (USER-WRITTEN SUBROUTINES GO HERE. SEE APPENDIX B)
7/8/9
SATELLITE ORBIT DETERMINATION USING AN EXTENDED KALMAN FILTER
 $PRDATA NF=4, NS=4, M=2, NZF=7, NZQ=2, ISEED=23, IPGSIZ=55,
 TF=5.0, DTPRPL=0.05, DTCCPL=0.05, DTMEAS=0.5, DTSTIX=0.5,
 IPASS=50, IPRRUN=1, LPRZR=.T., LPRLT=.T., LPP=.T., LCC=.T.,  $
1,2, 2,1, 2,4, 3,4, 4,1, 4,2, 4,4 / 2,2, 4,4
1.,0.,0.,1. /  1.,0.,0.,1.
1,1,.1 / 2,2,.1 / 3,3,.1 / 4,4,.1 / 0,0,.0
 $IN RFIN(1)=0.01,0.04,   QFIN(1)=2*0.02,   G0=1.0   $
1.0
1,1,1,1.
2,2,2,1.
3,3,3,57.2957795
4,4,4,57.2957795
0,0,0,1.
     TIME (TIME UNITS)
TRUE RANGE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3
     RANGE *LENGTH*
TRUE RANGE RATE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3
     RANGE RATE*LEN/UNIT TIME*
TRUE ANGLE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3
     ANGLE *DEGREES*
TRUE ANGLE RATE ERROR --> 1 : +SIGMA --> 2 : -SIGMA --> 3
     ANGLE RATE*DEG/UNIT TIME*
7/8/9
6/7/8/9
```

Figure D-2. Job Control, All User Input on Cards

REFERENCES

1. E. L. Hamilton, G.Chitwood and R.M.Reeves, "The General Covariance Analysis Program (GCAP), An Efficient Implementation of the Covariance Analysis Equations", Air Force Avionics Laboratory, WPAFB, Ohio, June 1976.

2. K. L. Jackson, "A Generalized Monte Carlo Analysis Program for Kalman Filter Design with Application to an Aircraft-to-Satellite Tracking Filter", Master's Thesis, AFIT/GGC/EE/77-6, December 1977.

3. N. A. Carlson, "Fast Triangular Formulation of the Square Root Filter", AIAA Journal, Vol. 11, No. 9, pp 1259-1265, September 1973.

4. V. N. Fadeeva, Computational Methods of Linear Algebra, Dover, New York, 1959.

5. R. E. Feldman and S. H. Musick, "SOFEPL: A Plotting Postprocessor for 'SOFE', User's Manual", AFWAL-TR-80-1109, Air Force Wright Aeronautical Laboratories, WPAFB, Ohio, to be published.

6. P. S. Maybeck, Stochastic Models, Estimation and Control, Volume 1, Academic Press, New York, 1979.

☆U.S.Government Printing Office: 1980 — 757-002/232

END

DATE FILMED

2-8

AD-A093 887 AIR FORCE WRIGHT AERONAUTICAL LABS WRIGHT-PATTERSON AFB OH F/G 9/2
SOFE: A GENERALIZED DIGITAL SIMULATION FOR OPTIMAL FILTER EVALU--ETC(U)
OCT 80 S H MUSICK
UNCLASSIFIED AFWAL-TR-80-1108 NL

3 of 3

AD
A093887

END
DATE
FILMED
3-82
DTIC

SUPPLEMENTARY

INFORMATION

AD-A093887

DEPARTMENT OF THE AIR FORCE
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES (AFSC)
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

REPLY TO ATTN OF: AAAN

8 Feb 82

SUBJECT: Errata for AFWAL-TR-80-1108 (AD A093887), SOFE User's Manual

TO: STINFO/TST

Please forward the attached errata page to DTIC and other agencies that were mandatory recipients of the named report. Original distribution occurred about January 1981.

Stanton H. Musick

STANTON H. MUSICK
Reference Systems Analysis and
Evaluation Group
Systems Avionics Division
Avionics Laboratory

1 Atch
Errata for AD A093887

1st Ind (TST/G. Doben/785-5572) 11 Feb 82

TO: Recipients

Attached is an errata sheet for AFWAL-TR-80-1108, "SOFE User's Manual."

JAMES G. JOHNSON, Director
Technical Information Center
Support Services Office

1 Atch
Errata Sheet

Errata for AD A093887
Feb 82

SOFE: A Generalized Digital Simulation for Optimal Filter Evaluation, User's Manual, Oct 1980

1. Page 25, third line from bottom. Correct this line to read

$$= E \{ (\underline{X}f-\hat{\underline{X}}f)\,(\underline{X}f-\hat{\underline{X}}f)^T \} \qquad \text{by (2-18)}$$

2. Page 46. Subroutine 'goplot' has been rewritten and now calls subroutine 'scale' for scaling of the y and t axes. 'scale' should be shown feeding into 'goplot' in the flowchart.

3. Page 48, lines 11 and 12 from top. These two lines note exceptions to ANSI standard practice that were employed 'in GOPLOT only'. The revised GOPLOT avoids these practices so these lines can be ignored.

4. Page 101, eq (5-7). Correct eq (5-7) to read

$$\ddot{r} = r\dot{\Theta}^2 - G_0/r^2 \qquad (5\text{-}7)$$

Note that the error being corrected here appeared in the SOFE manual but not in the computer code; therefore the satellite orbit results are correct as they appear in the manual.

5. Page 111, third line from bottom. New check product reflects correction of errata item 6.

-0.208788421550344 E-114

6. Page 116, subroutine FQGEN. Change F(1) to agree with eq (5-4).

F(1) = 1.

Note that correction 6 causes a substantial increase in the sigma of filter state 1 (at T=36000, old 2.88592, new 36,3497) and affects other states also but not nearly as much. The conclusion that the INS Kalman filter diverges is still correct.